VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Multiplatformní mobilní aplikace pro orientaci na VŠB-TU Ostrava

Multi-platform Mobile Application for VSB-TUO Contacts and Information Services

2018 Andrej Mikoláš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Andrej Mikoláš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Multiplatformní mobilní aplikace pro orientaci na VŠB-TU Ostrava
Multi-platform Mobile Application for VSB-TUO Contacts and Information Services

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce bude tvorba multiplatformní mobilní aplikace, využívající RESTful API, poskytnuté Centrem Informačních Technologií VŠB-TU Ostrava (CIT) k vyhledávání kontaktů v telefonním seznamu včetně informací o umístění místností a jednoduché navigace k nim včetně zobrazení polohy v mapě. Aplikace bude dále poskytovat doplňující informace o univerzitě a novinky.

1. Prostudujte existující aplikace podobného zaměření, používané v akademickém prostředí, a zjistěte současné služby poskytované CIT pro účely orientace na VŠB-TUO.
2. Zjistěte a popište současný stav API pro poskytování přístupu ke kontaktním údajům, orientaci v areálu a poskytování informací a novinek a navrhněte případná rozšíření.
3. Analyzujte a navrhněte aplikaci, zvolte vhodné implementační prostředí, popište knihovny, použité pro implementaci aplikace a jejich výběr zdůvodněte.
4. Implementujte multiplatformní řešení ve frameworku Apache Cordova nebo Xamarin.
5. Aplikaci otestujte alespoň na dvou platformách, a pokud to bude možné, vhodným způsobem publikujte. Výsledky testů vyhodnoťte.

Seznam doporučené odborné literatury:

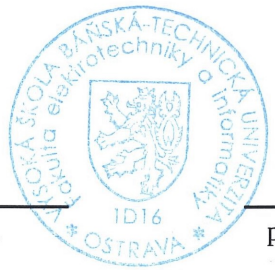
[1] Nilanchala Panigrahy: Xamarin Mobile Application Development for Android, Packt Publishing; 2. ed., 2015, 296 stran, ISBN: 978-1785280375.
[2] Dan Hermes: Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals, 2015, 436 stran, ISBN: 978-1484202159.
[3] Erik Hazzard: OpenLayers 2.10. Packt Publishing, 2011, 272 stran, ISBN: 978-1-84951-412-5.
[4] Charles Petzold: Creating Mobile Apps with Xamarin.Forms, Microsoft Press, 2016, 1162 stran, [online: https://aka.ms/xamebook].

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Moravec, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 20. duben 2018

.....

Na tomto mieste by som veľmi rád poďakoval pánovi Ing. Pavlovi Moravcovi, Ph.D. za odborné vedenie a ústretovosť počas konzultácii a vypracovávania tejto práce. Taktiež by som chcel poďakovať mojej rodine a známym za podporu počas práce na tomto zadaní.

Abstrakt

Cieľom tejto bakalárskej práce je návrh a implementácia multiplatformovej mobilnej aplikácie, využívajúcej služby RESTful API poskytnutého Centrom informačných technológii VŠB-TU Ostrava a tiež analýzou poskytovaných služieb. Súčasne sa práca zaoberá problematikou práce s RESTful API a problematikou vývoja multiplatformových mobilných aplikácii. Práca sa ďalej zaoberá prieskumom súčasného trhu s mobilnými platformami a tiež prieskumom existujúcich aplikácii podobného zamerania využívaných v akademickom prostredí. Výstupom práce je funkčná aplikácia umožňujúca orientáciu v areáli VŠB-TUO, vyhľadávanie kontaktných údajov na zamestnancov univerzity a poskytovanie informácii o univerzitných novinkách.

Kľúčové slová: mobilná aplikácia, Xamarin, Android, iOS, REST, navigácia, kontakty

Abstract

The purpose of this bachelor thesis is to design and implement a multiplatform mobile application which uses the RESTful API services provided by the VŠB-TU Ostrava Information Technology Center as well as an analysis of provided services. Thesis also deals with the issue of RESTful API and the issue of development of multiplatform mobile applications. The thesis is also focused on exploring the current market for mobile platforms as well as exploring existing applications of a similar focus used in the academic environment. The output of the bachelor thesis is a functional application allowing orientation in VŠB-TUO grounds, searching of contact data for university employees and providing information about university news.

Key Words: mobile application, Xamarin, Android, iOS, REST, navigation, contacts

Obsah

Zoznam použitých skratiek a symbolov

API	–	Application Programming Interface (rozhranie pre programovanie aplikácii)
CIT	–	Centrum informačných technológii VŠB-TU Ostrava
HTTP	–	Hypertext Transfer Protocol (hypertextový prenosový protokol)
IS	–	Informačný systém
JSON	–	JavaScript Object Notation (JavaScriptový objektový zápis)
OS	–	Operačný systém
QR	–	Quick Response (rýchla odpoveď)
REST	–	Representational State Transfer
SSH	–	Secure Shell (zabezpečený prístup k príkazovému interpretovaču)
URI	–	Uniform Resource Identifier (jednotný identifikátor zdroja)
URL	–	Uniform Resource Locator (jednotný vyhľadávač prostriedku)
XAML	–	Extensible Application Markup Language (rozšíriteľný aplikačný značkovací jazyk)
XML	–	Extensible Markup Language (rozšíriteľný značkovací jazyk)

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

Úvod

Mobilný telefón je dnes bežnou výbavou každého človeka. Posledné štatistiky hovoria o viac než piatich miliardách mobilných účastníkov[1]. Ich hlavným účelom už dávno nie je len telefonovanie. Siahame po nich za účelom zábavy, organizácie dňa, spájaniu sa s okolitým svetom pomocou sociálnych sietí alebo ich využívame ako pracovný nástroj. Práve vďaka získaniu obľúbenosti medzi užívateľmi, zaznamenal za posledných desať rokov trh s mobilnými telefónmi významný technologický posun vpred. Od jednoduchých zariadení určených na telefonovanie sa výrobcovia prepracovali až ku multifunkčnému, technologicky vyspelému výpočtovému zariadeniu so širokou škálou využitia. Neustále zvyšovanie výkonu a výdrže batérie, vyvíjanie pokročilejších technológii, pridávanie nových zabezpečovacích prvkov, vytváranie kvalitnejších fotografií či prepracovanejšie displeje sú hlavnými hnacími motormi pre ďalší vývoj.

Ako každé výpočtové zariadenie, aj mobilný telefón potrebuje pre svoju činnosť operačný systém. Bez neho je dané zariadenie nepoužiteľné. Výrobcovia mobilných operačných systémov do nich implementujú čoraz inovatívnejšie funkcie, ktorými sa snažia v konkurenčnom boji získať zákazníka na svoju stranu.

Mobilné aplikácie obohacujú základný operačný systém o doplňujúcu funkcionalitu. Aplikácie môžu slúžiť na vzdelávanie, zábavu, relax alebo ako užitočné nástroje v rôznych oblastiach. Keďže si aplikácie inštalujeme do mobilných telefónov, ktoré máme neustále pri sebe, máme teda na dosah ruky aj dané aplikácie. Tento fakt je možné využiť a poskytnúť študentom "do vrecka" užitočné informácie, ktoré sú súčasťou ich štúdia. Informácie, ktoré by mohli byť pre študentov tejto univerzity užitočné, poskytuje Centrum informačných technológií VŠB-TUO prostredníctvom svojich webových stránok alebo RESTful API. Riešením je teda mobilná aplikácia pre študentov, poskytujúca rýchly a jednoduchý prístup k týmto dátam.

Úvod práce bude venovaný súčasnému stavu trhu s mobilnými platformami. Bude uvedený prieskum zastúpenia mobilných platforiem a u najrozšírenejších z nich aj zastúpenie ich jednotlivých verzii. Ďalej popíšem aké sú súčasné možnosti vývoja mobilných aplikácii a aké sú techniky vývoja aplikácii pre viac platforiem.

V nasledujúcej kapitole popíšem existujúce aplikácie, ktoré využívajú študenti na ostatných univerzitách a tiež aplikácie, ktoré boli vyvinuté pre študentov VŠB-TUO. Tento prieskum je dôležitý kvôli zisteniu, akú funkcionalitu by mala poskytovať dobrá študentská aplikácia a zároveň, či už momentálne neexistujú hotové riešenia pre túto univerzitu.

V ďalšej kapitole analyzujem súčasné služby pre orientáciu v areáloch a budovách, ktoré poskytuje CIT pre študentov a návštevníkov univerzity. Nasledujúca kapitola zoznámi čitateľa s problematikou RESTful API a tiež analyzuje súčasný stav a služby RESTful API poskytovaného CIT. Bude uvedená základná charakteristika a obsah daných služieb a navrhnuté budú prípadné rozšírenia.

[1] `https://www.gsmaintelligence.com/`

Nasledujúca kapitola sa venuje návrhu aplikácie. Návrh bude popísaný formou určitých logických krokov, ktoré postupne vykreslia finálnu podobu aplikácie, ktorá je predmetom tejto bakalárskej práce. Návrh bude doplnený o potrebné diagramy, ktoré jasnejšie popíšu systém z daného uhla pohľadu.

Predposlednú kapitolu predstavuje samotná implementácia aplikácie. Jedná sa o najdôležitejšiu kapitolu tejto práce. Popíše, ako boli implementované vybrané časti aplikácie, aké nástroje boli počas vývoja použité a tiež, ktoré knižnice boli zakomponované pre dosiahnutie požadovanej funkcionality.

Záverečná kapitola práce sa venuje procesu testovania. Kapitola popíše testovanie RESTful API CIT a tiež samotnej aplikácie. Uvedený bude tiež spôsob, ako môžu užívatelia nahlasovať prípadné zistené nedostatky aplikácie a tiež akým spôsobom bude aplikácia sama zdieľať údaje o chybách.

1 Prieskum trhu

V úvode práce sa zameriam na súčasný stav a zastúpenie operačných systémov pre mobilné telefóny. Pre platformy s významným zastúpením uvediem tiež aktuálne zastúpenie ich jednotlivých verzii na trhu. Ďalej popíšem, aké sú možnosti vývoja aplikácii pre najrozšírenejšie platformy a tiež aké technológie sa v súčasnosti používajú na vývoj multiplatformových aplikácii.

1.1 Mobilné platformy

Na rozdiel od operačných systémov pre osobné počítače, ten mobilný je prispôsobený pre beh na hardvéri s obmedzeným výkonom, veľkosťou obrazovky alebo zdrojom napájania, ktorý predstavuje napäťový akumulátor. Preto musí oveľa efektívnejšie narábať s pamäťovými prostriedkami, zobrazovať obsah na menšej obrazovke a tiež sa snažiť o čo najnižšiu spotrebu napájania.

Niektorí výrobcovia mobilných telefónov si vyvíjajú aj vlastný operačný systém. Ostatní využívajú pre svoje zariadenia operačný systém od iného výrobcu, ktorý si však optimalizujú alebo upravia podľa vlastných predstáv. Pre využitie plného potenciálu systému ponúkajú výrobcovia OS aj digitálne distribučné platformy, kde vývojári môžu predávať svoje aplikácie. Cieľový zákazník sa teda pohodlne dostane k rozširujúcej softvérovej funkcionalite svojho mobilného telefónu.

1.1.1 Zastúpenie na trhu

Na grafe na obrázku 1 môžeme vidieť celosvetové štatistiky používaných operačných systémov na mobilných zariadeniach za obdobie január až december 2017. Z grafu vyplýva, že najviac rozšírené sú Android od spoločnosti Google a iOS od spoločnosti Apple. V minulosti mal určitý značný podiel na trhu aj operačný systém Windows Phone od spoločnosti Microsoft, no jeho zastúpenie v dôsledku nezáujmu verejnosti v posledných rokoch prudko klesalo, čo nakoniec viedlo až k úplnému ukončeniu podpory pre tento systém zo strany výrobcu.

Android

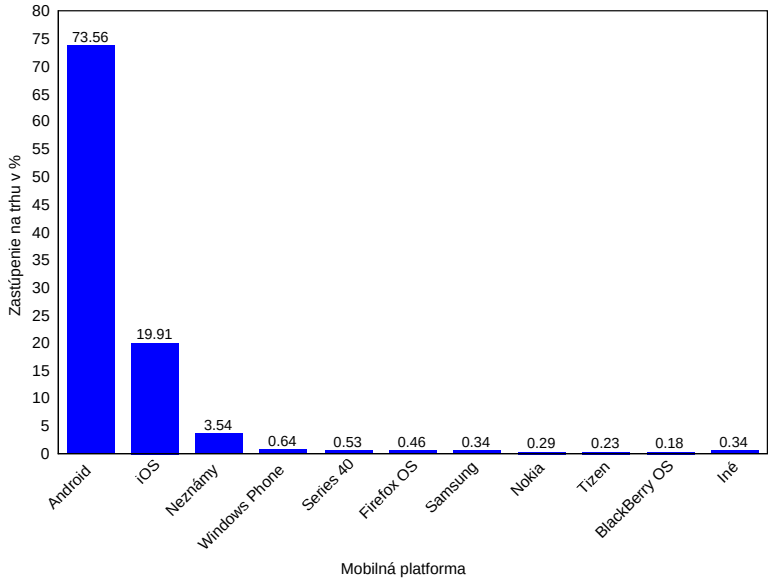
Android predstavuje najrozšírenejší a najobľúbenejší mobilný operačný systém súčasnosti. Jedná sa o *Open Source* platformu vyvíjanú spoločnosťou Google. Práve vďaka otvorenosti systému si ho každý výrobca mobilných telefónov môže obohatiť o vlastnú nadstavbu, čím dosiahne odlišné užívateľské rozhranie alebo doplňujúcu funkcionalitu. Obľúbenosti tejto platformy medzi užívateľmi pridáva aj možnosť inštalovať z viac než troch miliónov[3] aplikácii dostupných vo virtuálnom obchode *Google Play Store*. [10]

iOS

Tento operačný systém vyvíja firma Apple. Na rozdiel od OS Android je iOS uzavretá platforma

[2] `http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201712-201712-bar`

[3] `https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/`



Obr. 1: Celosvetové zastúpenie mobilných OS za december 2017[2]

a výrobca ho využíva len na svojich mobilných telefónoch iPhone, tabletoch iPad a hudobných prehrávačoch iPod Touch. Medzi najväčšie prednosti systému patrí jeho jednoduchosť, prehľadnosť, výkonnosť a v neposlednom rade aj podpora zo strany výrobcu vo forme pravidelných aktualizácii. Možnosť sťahovania doplnkových aplikácii je zabezpečená pomocou distribučnej platformy *Apple App Store*. [10] V prípade iOS však platia prísnejšie opatrenia a podmienky kladené na aplikácie, ktoré vývojári môžu publikovať v tomto obchode.

1.1.2 Zastúpenie verzii významných operačných systémov

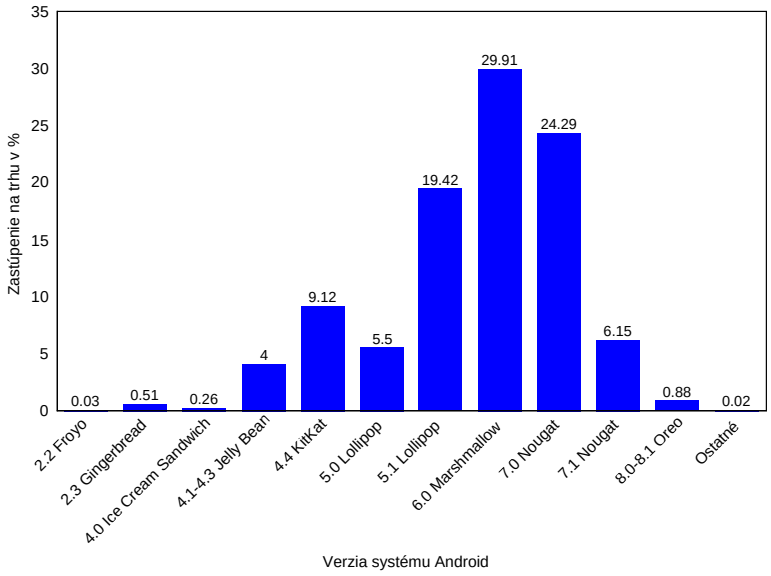
Pri výbere cieľových platforiem pre novú mobilnú aplikáciu hrá dôležitú úlohu aj najnižšia podporovaná verzia danej platformy. Aplikácia musí podporovať tak staré verzie, aby bola pokrytá čo najväčšia cieľová skupina užívateľov, avšak nie až tak staré, aby sme prišli o podporu moderných funkcionalít, ktoré sú dostupné len pri novších verziách.

Android

V prípade OS Android je distribúcia nových verzii pomalšia než v prípade iOS. Najnovšie verzie systému sa výrobcovia snažia primárne zaviesť do svojich najnovších a najvýkonnejších zariadení, prípadne ich plánujú inštalovať do pripravovaných, zatiaľ nevydaných modelov. Staršie zariadenia sa k aktualizácii dostanú neskôr alebo vôbec, čím výrobcovia značne ovplyvňujú percentuálne zastúpenie najnovšej verzie systému na trhu. Preto je v prípade tejto platformy nutné zameriavať sa aj na staršie verzie.

Z grafu na obrázku 2 je zrejmé, že v súčasnosti najrozšírenejšou verziou je verzia 6.0, avšak aktuálne najnovšou verziou je 8.1. Z grafu je tiež zrejmé, že najstaršou verziou, ktorá má ešte značný podiel na trhu, je verzia 4.4 s kódovým označením KitKat a odpovedajúcim API 19.

Táto verzia bola vydaná v septembri 2013. Je teda vhodné, aby v súčasnosti vyvíjané aplikácie boli spätne kompatibilné aj s touto verziou.



Obr. 2: Celosvetové zastúpenie verzii OS Android na mobilných telefónoch ku dňu 16.1.2018[4]

iOS

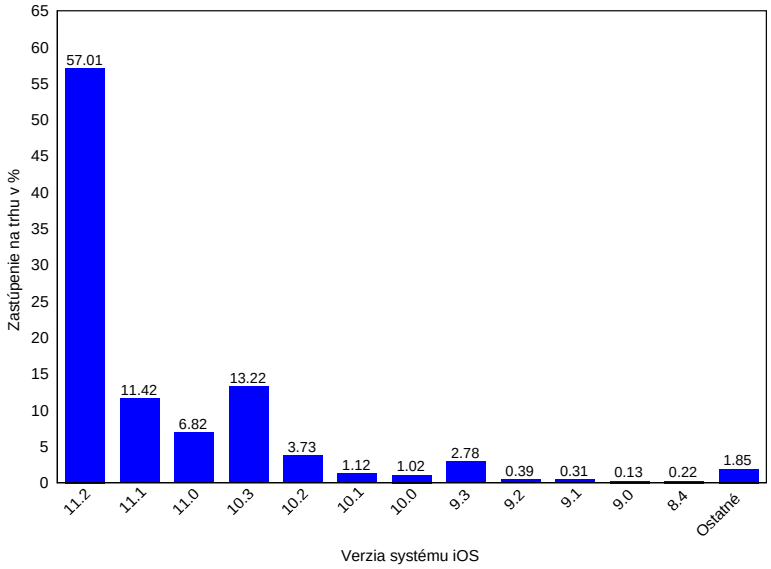
Ako môžeme vidieť na obrázku 3, u operačného systému iOS majú väčšinové zastúpenia práve najnovšie verzie systému. Je to najmä tým, že Apple pri vydaní aktualizácie sa zameriava nie len na najnovší model zo svojej rady, ale poskytuje podporu aj pre zariadenia o niekoľko generácii staršie. Aktualizácia je pritom dostupná pre všetky podporované zariadenia naraz. V porovnaní s OS Android je to značný rozdiel, no dôvodom je rozmanitosť zariadení, pre ktoré je potrebné najnovší systém upraviť a každý výrobca si úpravu musí zabezpečiť sám. Jedná sa o stovky Android zariadení, na ktorých by musela byť najnovšia verzia vyladená, no v prípade iOS sa jedná o menej než desať modelov.

1.2 Možnosti vývoja mobilných aplikácii

Spôsob implementácie mobilnej aplikácie musí byť stanovený už v rannej fáze návrhu systému. Dôležitým faktorom pri výbere je určenie, či chceme aplikáciu pre jednu alebo viacero platforiem. Následne príde na rad rozhodnutie, či sa uplatní natívny alebo multiplatformový prístup. Rozhodnutie môžu tiež ovplyvniť súčasné možnosti vývojára či vývojárov, časová alebo finančná náročnosť. Nasleduje popis jednotlivých prístupov spolu s ich výhodami a nevýhodami.

[4] `http://gs.statcounter.com/android-version-market-share/mobile/worldwide/#daily-20180116-20180116-bar`

[5] `http://gs.statcounter.com/ios-version-market-share/mobile/worldwide/#daily-20180116-20180116-bar`



Obr. 3: Celosvetové zastúpenie verzii iOS na mobilných telefónoch ku dňu 16.1.2018[5]

1.2.1 Natívna aplikácia

Natívna aplikácia je vyvinutá pre konkrétnu platformu a nie je možné ju spustiť na inej. Vyznačujú sa vyšším výkonom a spoľahlivosťou než multiplatformové. Sú vhodným riešením pre aplikácie vyžadujúce extrémne rýchlu odozvu a poskytnutie vysokého výkonu. Príkladom môžu byť napríklad hry alebo aplikácie pracujúce s veľkým objemom dát. Na obrázku 4 je možné vidieť, že pri vývoji natívnej aplikácie je nutné opakovať proces implementácie a testovania pre každú platformu osobitne. Po následnej kompilácii dostaneme inštalačný súbor pre konkrétnu platformu.

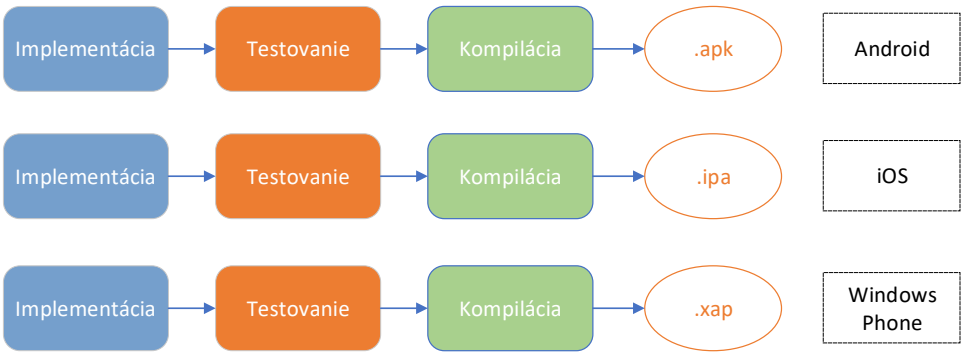
Výhody

- Vyšší výkon aplikácii
- Väčšie využitie plného potenciálu danej platformy
- Lepší prístup ku natívnym funkciám a ovládacím prvkom danej platformy
- Možnosť lepšie využiť grafické ovládacie prvky danej platformy a tým docieliť prirodzenejší a jednotnejší vzhľad aplikácii

Nevýhody

- Nutnosť vyvíjať aplikáciu od začiatku pre každú platformu (s výnimkou zdieľaných knižníc, ktoré je možné použiť opakovane)
- Nutnosť znalosti viacerých programovacích jazykov

- Dlhší čas strávený vývojom
- Vyššie náklady na vývoj pre viacero platforiem



Obr. 4: Postup vývoja natívnej aplikácie pre viaceré platformy

1.2.2 Hybridná multiplatformová aplikácia

Pri vytváraní softvéru je výhodnejšie uplatňovať multiplatformový prístup. To znamená, že vytvorený program (resp. napísaný kód) je možné spustiť na viacerých počítačových platformách. Za platformu sa považuje kombinácia hardvéru a softvéru, ktorá je nutná pre spustenie softvérového produktu.

Hybridné mobilné aplikácie využívajú spoločný kód, tvoriaci napríklad biznis logiku, prístup do databázy, prístup ku webovým službám alebo prístup ku špecifickému hardvéru ako napríklad fotoaparát, GPS modul alebo zvukové zariadenia. Užívateľské rozhranie môže byť tvorené taktiež spoločným kódom, alebo v závislosti od použitého frameworku sa môže uplatniť natívny prístup. Iba malá časť kódu je tvorená pre každú platformu zvlášť. Diagram na obrázku 5 ukazuje, že pri vývoji multiplatformovej aplikácie nie je nutné opakovať proces implementácie a testovania pre každú platformu zvlášť. Proces kompilácie sa však opakovať musí, nakoľko pri nej dochádza k prekladu kódu do natívneho jazyka danej platformy.

Môžu pracovať aj bez internetového pripojenia a bežný užívateľ nemá možnosť spoznať, že pracuje s hybridnou a nie natívnou aplikáciou. Ich distribúcia je zabezpečená digitálnymi distribučnými platformami ako napríklad *Google Play Store* v prípade OS Android alebo *Apple App Store* v prípade iOS.

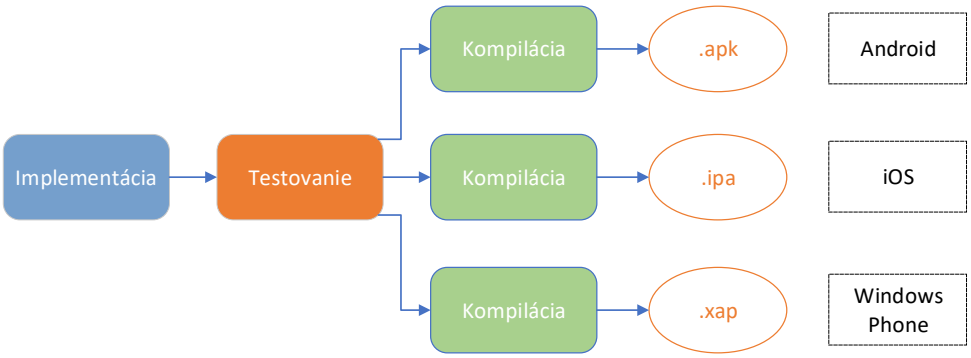
Výhody

- Jeden zdrojový kód zdieľaný naprieč platformami
- Využitie jedného frameworku – potreba znalosti menej programovacích jazykov
- Počas vývoja vznikajú verzie pre viaceré platformy naraz
- Šetrenie času vývoja

- Šetrenie nákladov na vývoj

Nevýhody

- Mierne nižší výkon – nevhodné napríklad pre hry
- Zložitejšia implementácia natívnych funkcii danej platformy
- Pozorný užívateľ môže spozorovať nie natívny pocit z užívateľského rozhrania



Obr. 5: Postup vývoja hybridnej multiplatformovej aplikácie

1.2.3 Webová multiplatformová aplikácia

Druh aplikácii, ktoré sa neinštalujú do zariadenia, ale sú spustené iba v prostredí internetového prehliadača. Majú teda obmedzený prístup ku hardvéru, v porovnaní s aplikáciou inštalovanou priamo do zariadenia. Sú optimalizované pre mobilné zariadenia, čo znamená, že neposkytujú taký výkon, ako aplikácie inštalované priamo v zariadení. Ich šírenie je zabezpečené len pomocou verejne dostupných URL adries. Ich hlavnou nevýhodou je, že sú spustiteľné iba v režime online.

1.3 Multiplatformový vývoj

Frameworky dostupné pre vývoj hybridných multiplatformových mobilných aplikácii predstavujú v súčasnosti napríklad Apache Cordova alebo Xamarin. Líšia sa napríklad použitými programovacími jazykmi alebo poskytovanou funkcionalitou.

1.3.1 Apache Cordova

Využívajú sa jazyky HTML5 a CSS pre užívateľské rozhranie a JavaScript pre logiku. Platformy, ktoré tento framework podporuje sú iOS, Android, Windows Phone a BlackBerry. Natívny vzhľad aplikácie sa implementuje veľmi obtiažne a len za pomoci rozširujúcich modulov. Znovupoužiteľnosť spoločného kódu dosahuje vysokej úrovne. Paralelné programovanie je možné len pomocou modulov a práca s väčším objemom dát spôsobuje vysokú odozvu aplikácie. Výkon aplikácie môže byť na starších zariadeniach obmedzený, no na novších sa dosahuje vysoký výkon. Plný prístup ku API zariadenia je možný pomocou rozširujúcich modulov. [25]

1.3.2 Xamarin

Tento framework, založený na platforme *Mono*, využíva programovací jazyk *C#* pre logiku a značkovací jazyk *XML* pre užívateľské rozhranie. [26] Kód je následne prekompilovný do natívneho jazyku konkrétnej platformy. Podporované cieľové platformy sú iOS, Android a Windows Phone. Dovoľuje vyvíjať aplikácie s natívnym užívateľským rozhraním a tiež plne pristupovať ku API daného zariadenia. Užívateľské rozhranie je možné tvoriť plne natívnym prístupom, teda využitím špecifického kódu pre konkrétnu platformu, alebo využiť nástroj *Xamarin Forms*, ktorým vytvoríme rozhranie naraz pre všetky platformy. Pri vývoji sa využíva až 80% spoločného kódu pre všetky cieľové platformy. [25] Z dôvodu použitia jazyku *C#*, podporuje tento framework tiež možnosť paralelného a asynchrónneho programovania a prácu s veľkým objemom dát. [27] Pri správnej implementácii je možné dosiahnuť vysokého výkonu.

1.4 Zhrnutie prieskumu

Z vykonaného prieskumu trhu som usúdil, že v súčasnosti najrozšírenejšími platformami pre mobilné telefóny sú OS Android a iOS. Ich rozšírenosť je stále na vysokej úrovni a preto je výhodnejšie sa pri vývoji zameriavať práve na tieto dve platformy. Čo sa týka vývoja pre viaceré platformy naraz, je výhodnejšie pri výkonovo nenáročných aplikáciách sa zamerať na hybridný multiplatformový vývoj. Tým môžeme výrazne znížiť čas a náklady na vývoj.

2 Súčasné aplikácie využívané v akademickom prostredí

V tejto časti práce popíšem aktuálne existujúce aplikácie, ktoré sa používajú na iných univerzitách a tiež aplikácie určené pre študentov VŠB-TUO.

2.1 Aplikácie využívané na iných univerzitách

Nasleduje popis stavu a funkcionality mobilných aplikácii, používaných na iných univerzitách v Česku a na Slovensku, ktoré sú dostupné na digitálnych distribučných platformách vybraných operačných systémov. Ku každej aplikácii uvediem, pre ktorú mobilnú platformu sú určené a zhrniem jej základné funkcie.

Všetky aplikácie, ktoré som skúmal vyžadujú pre prístup ku všetkým funkciám prihlásenie pomocou univerzitného účtu študenta, ktorý však nemám, nakoľko nie som študentom týchto univerzít. Preto budem pri posudzovaní ich funkcii brať do úvahy popis a priložené snímky obrazoviek zverejnené na digitálnych distribučných platformách *Google Play Store* alebo *Apple App Store*.

2.1.1 UPlikace

Mobilná platforma: Android (verzia 4.0 a vyššia)
Popisovaná verzia aplikácie: 1.3.3
Dostupné funkcie (spracované podľa [1]):

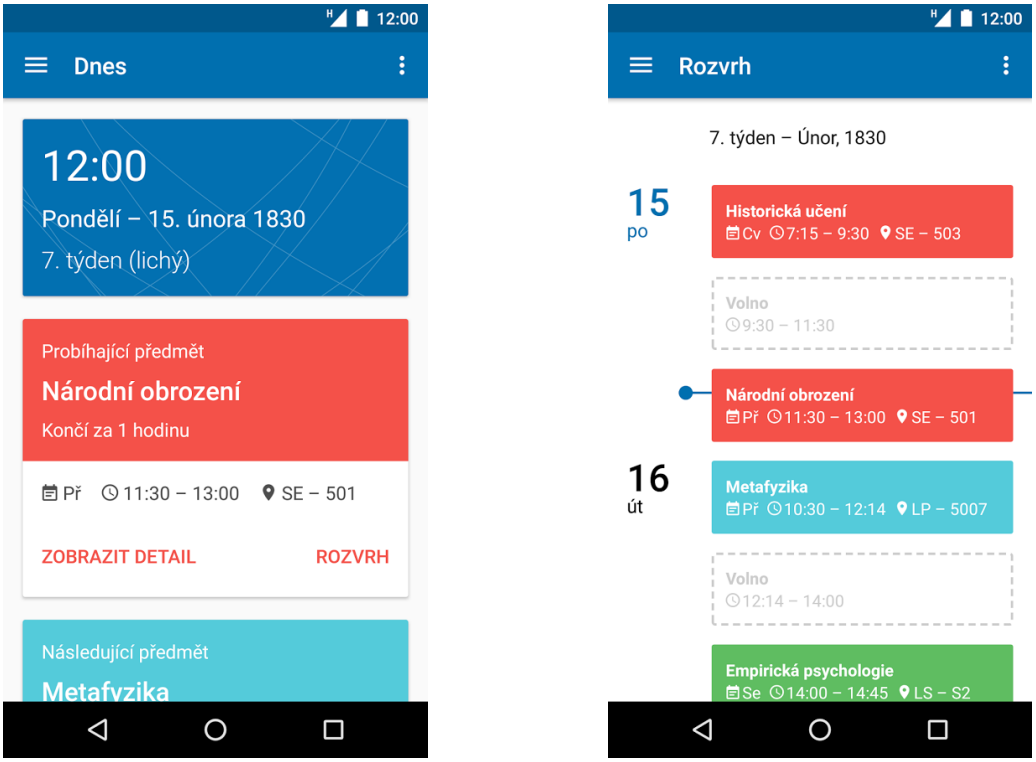
- Detailný prehľad o štúdiu
- Rozvrh
- Harmonogram skúškových termínov
- Interaktívna mapa mesta Olomouc
- Možnosť prihlasovania a odhlasovania sa z termínov skúšky
- Notifikácie pri vložení známky do univerzitného systému a pri uvoľnení miesta na zaplnenom skúškovom termíne

Táto aplikácia je určená pre študentov Univerzity Palackého v Olomouci. Užívateľské rozhranie je tvorené v modernom štýle *Material Design*. Aplikácia je rozdelená na záložky, medzi ktorými sa užívateľ presúva posunutím cez obrazovku.

Jednotlivé záložky sú:

- **Dnes:** Zobrazuje informácie o dátume, čase, poradí týždňa a to či je aktuálny týždeň párny alebo nepárny. Ďalej záložka obsahuje rozpis výuky pre konkrétny deň. Ukážka záložky je na obrázku 6 vľavo.

- **Rozvrh:** Je veľmi prehľadne zobrazovaný ako zoznam jednotlivých nasledujúcich vyučovacích hodín, medzi ktorými je aj informácia, koľko voľného času študent má do nasledujúcej výuky. Ukážka je na obrázku 6 vpravo.
- **Predmety:** Zoznam všetkých aktuálnych predmetov študenta s farebne rozlíšenými ikonami.
- **Skúšky:** Je rozdelená na dve pod záložky:
  - **Výsledky:** Obsahuje zoznam všetkých aktuálnych predmetov študenta a stav ich dokončenia. Je uvedený stav zápočtu a stav skúšky.
  - **Termíny:** Zobrazuje jednotlivé nasledujúce termíny skúšok v chronologickom poradí. Pravdepodobne po kliknutí na termín bude študent prihlásený.
- **Mapa:** Zobrazuje výsek oblasti z Google Maps, kde sú zaznačené dôležité body v areáli univerzity.



Obr. 6: Uplikace – ukážka užívateľského rozhrania [1]

2.1.2 Muni IS

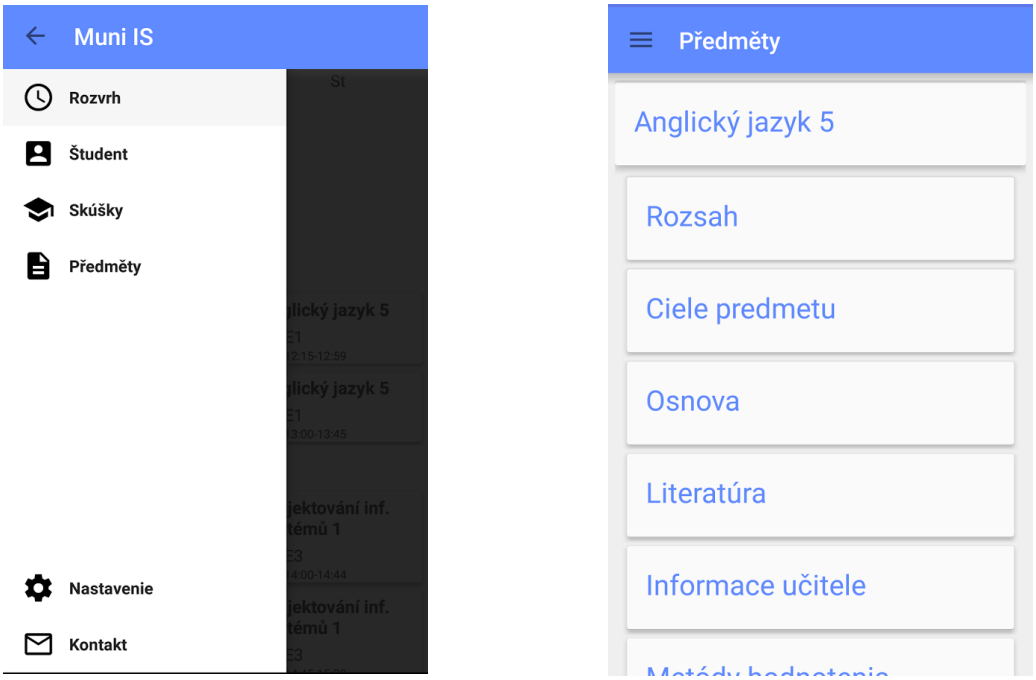
Mobilná platforma: Android (verzia 4.1 a vyššia)
Popisovaná verzia aplikácie: 2.07

Dostupné funkcie (spracované podľa [2]):

- Rozvrh pre aktuálny a nasledujúci týždeň
- Študijné novinky, poznámkové bloky
- Známky
- Termíny skúšok
- Študijné materiály

Aplikácia je určená pre študentov vysokých škôl, ktoré využívajú informačný systém vyvíjaný na Masarykovej univerzite. Jedná sa napríklad o vysoké školy: Janáčkova akademie múzických umění, Vysoká škola technická a ekonomická alebo Masarykova univerzita. Táto aplikácia však nie je oficiálnym dielom Masarykovej univerzity.

Pri práci s aplikáciou dochádza len ku čítaniu dát z IS, zmena dát nie je možná. To znamená, že nie je možné napríklad prihlasovanie na skúškové termíny. Na obrázku 7 vľavo je ukážka vysúvacieho menu, kde sa užívateľ prepína medzi jednotlivými obrazovkami aplikácie. Obrázok 7 vpravo zobrazuje obrazovku s detailom predmetu.



Obr. 7: MuniIS – ukážka užívateľského rozhrania [2]

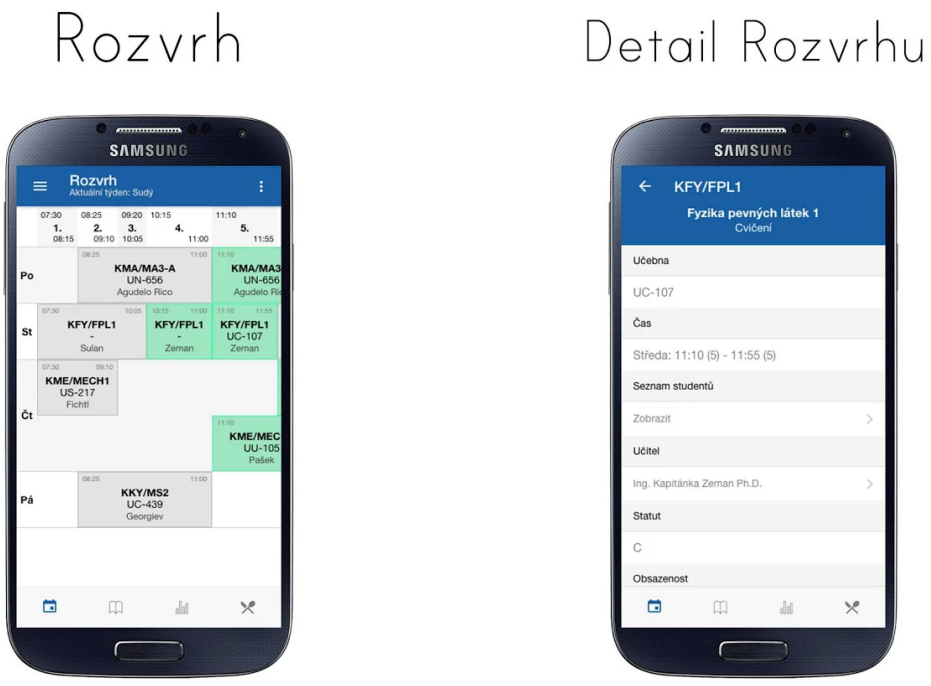
2.1.3 Student ZČU

Mobilná platforma: Android (verzia 4.4 a vyššia), iOS (verzia 8.0 a vyššia)

Popisované verzie aplikácií: Android - 1.4.4, iOS - 1.4.2
Dostupné funkcie (spracované podľa [3]):

- Rozvrh
- Prihlasovanie a odhlasovanie sa z termínov skúšky
- Študijné výsledky
- Jedálny lístok na nasledujúcich 5 dní v stravovacích zariadeniach univerzity

Aplikácia je určená pre študentov Západočeskej univerzity v Plzni. Užívateľské rozhranie je tvorené pomocou štyroch základných obrazoviek, na ktorej sú jednotlivé spomínané funkcie. Medzi obrazovkami sa užívateľ prepína pomocou ikoniek v dolnej časti obrazovky. Ako môžeme vidieť na obrázku 8 vľavo, rozvrh je zobrazený klasickou formou, kde je na jedenej obrazovke zobrazený rozvrh na celý týždeň. Pravdepodobne po kliknutí na konkrétnu hodinu v rozvrhu bude užívateľ presmerovaný na obrazovku zobrazenú na obrázku 8 vpravo, kde sú detailnejšie informácie. Na obrazovke s výsledkami je zoznam predmetov študenta s farebne rozlíšenými splnenými a nesplnenými podmienkami predmetu.



Obr. 8: Student ZČU – ukážka užívateľského rozhrania [3]

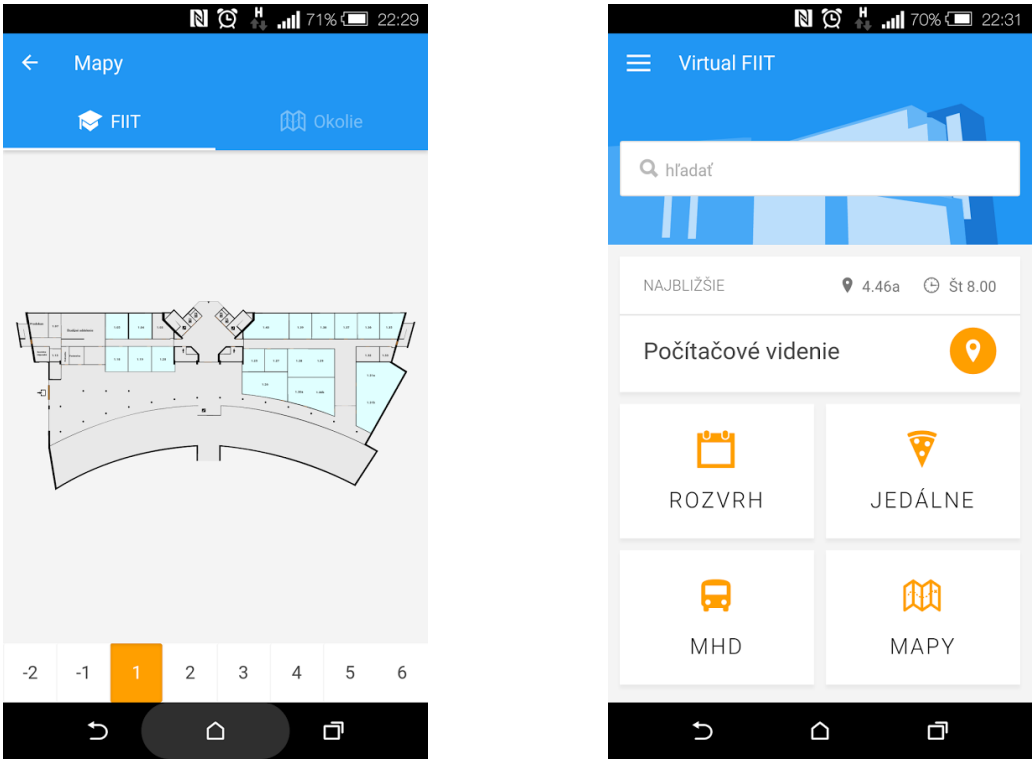
2.1.4 Virtual FIIT

Mobilná platforma: Android (verzia 4.0 a vyššia)

Popisovaná verzia aplikácie: 2.0.11
Dostupné funkcie (spracované podľa [4]):

- Interaktívna mapa všetkých poschodí budovy univerzity (ukážka je na obrázku 9 vľavo)
- Rozpis učební a ku každej jej rozvrh
- Cestovný poriadok mestskej hromadnej dopravy zo zastávok v blízkosti univerzity
- Jedálny lístok v stravovacích zariadeniach univerzity
- Interaktívny rozvrh
- Odkazy na dôležité webové stránky súvisiace so štúdiom na fakulte
- Informácie o učiteľoch, učebniach a predmetoch
- Harmonogram semestra

Aplikácia je určená pre študentov Fakulty informatiky a informačných technológii Slovenskej technickej univerzity v Bratislave. Užívateľské prostredie pôsobí veľmi príjemne, moderne a prehľadne. Na úvodnej obrazovke zobrazenej na obrázku 9 vpravo je informácia o najbližšej nadchádzajúcej výuke a rýchly prístup ku rozvrhu, jedálnemu lístku, spojom mestskej hromadnej dopravy a ku mape.



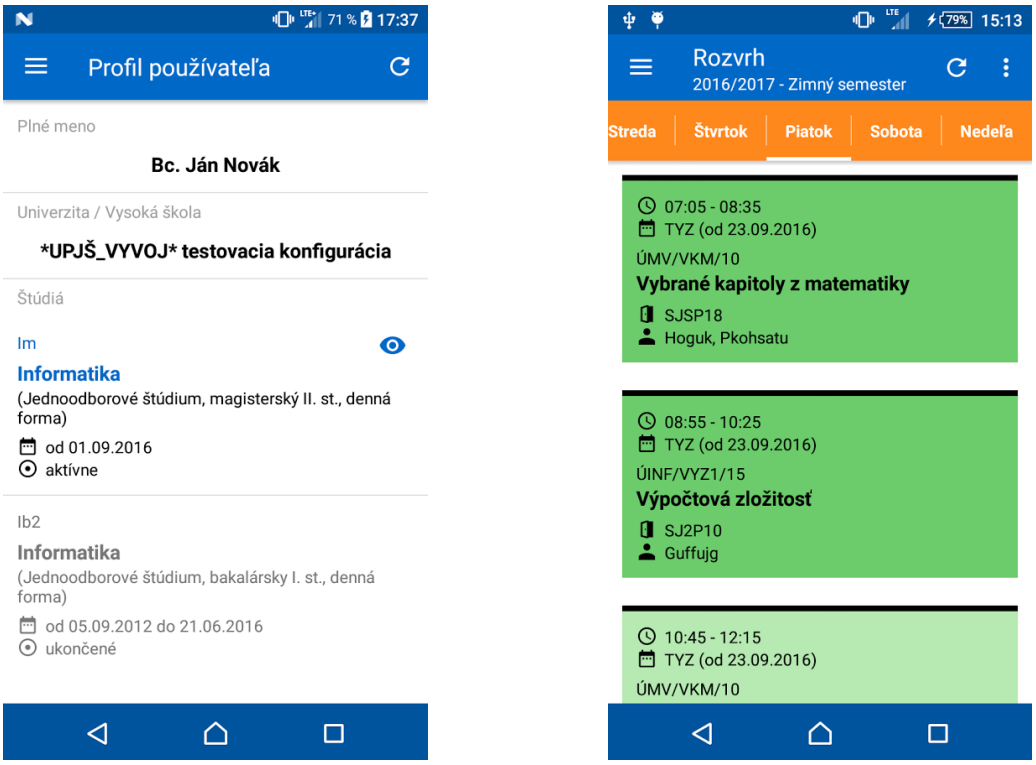
Obr. 9: Virtual FIIT – ukážka užívateľského rozhrania [4]

2.1.5 AiS2 študent

Mobilná platforma: Android (verzia 3.0 a vyššia)
Popisovaná verzia aplikácie: 2.3.6
Dostupné funkcie (spracované podľa [5]):

- Prihlasovanie a odhlasovanie sa z termínov skúšky
- Výsledky jednotlivých skúškových termínov
- Študijné výsledky
- Rozvrh (ukážka je na obrázku 10 vpravo)

Užívateľské rozhranie pôsobí celkom prehľadne. Aplikácia tiež obsahuje profil študenta, kde sú uvedené kompletné informácie o jeho aktuálnom, ale aj predchádzajúcom štúdiu. Ukážka profilu je na obrázku 10 vľavo.



Obr. 10: AiS2 študent - ukážka užívateľského rozhrania [5]

2.2 Aplikácie vyvinuté pre VŠB-TUO

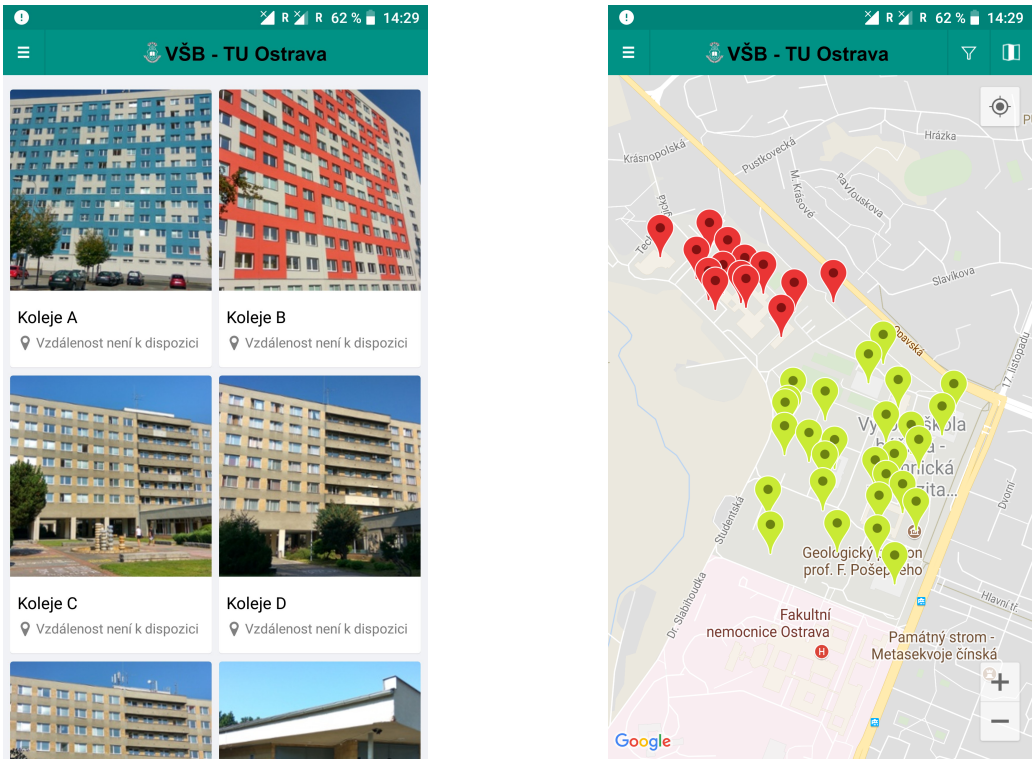
V tejto podkapitole popíšem aplikácie, ktoré sú vyvinuté pre študentov tejto univerzity a sú zároveň dostupné na online distribučných platformách.

2.2.1 Průvodce VŠB-TU Ostrava

Mobilná platforma: Android (verzia 2.3.3 a vyššia), iOS (verzia 7.0 a vyššia)
Popisované verzie aplikácií: Android - 1.3.3 (zo dňa 5.11.2016), iOS - 8 (zo dňa 6.11.2016)
Dostupné funkcie (spracované podľa [6]):

- Popis miest v jednotlivých areáloch s možnosťou zobrazenia na mape a navigácie ku nim
- Kontaktné informácie na univerzitu
- Užitočné odkazy pre študentov

Aplikácia na mňa po vizuálnej stránke nepôsobila veľmi prívetivo. Na obrázku 11 vľavo je zobrazená úvodná obrazovka s miestami. Vyvinutá bola multiplatformovým prístupom pomocou nástroja *Effortix*[6], ktorý umožňuje vyvíjať mobilné aplikácie bez znalosti programovania. Užívateľské rozhranie vyzerá na oboch podporovaných platformách úplne identicky, takže sa z aplikácie úplne vytráca natívny dojem. Aplikácia vracia relevantné údaje, no obsahuje tiež zbytočné funkcie, ako napríklad skener QR kódov alebo možnosť rozšírenej reality. Pre zobrazovanie miest a navigáciu k nim sa využívajú Mapy Google, zobrazené na obrázku 11 vpravo.



Obr. 11: Průvodce VŠB-TU Ostrava - ukážka užívateľského rozhrania [6]

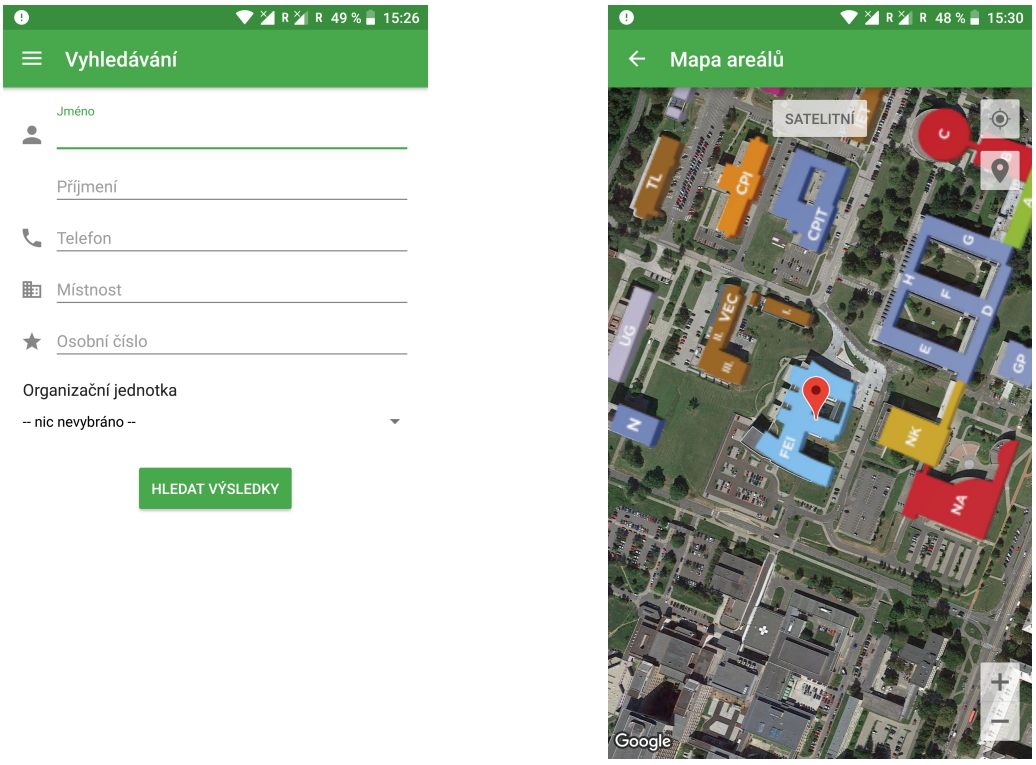
[6] `https://www.effortix.com/?ref=lwi`

2.2.2 VŠB Telefonní Seznam

Mobilná platforma: Android (verzia 4.0.3 a vyššia)
Popisovaná verzia aplikácie: 1.0 (zo dňa 2.5.2016)
Dostupné funkcie (spracované podľa [7]):

- Vyhľadávanie kontaktov v adresári
- Univerzitné novinky
- Informácie o univerzite
- Navigácia do miestností

Užívateľské rozhranie aplikácie je veľmi prehľadné a jednoduché. Pre vyhľadávanie kontaktov slúži jednoduchý formulár, zobrazený na obrázku 12 vľavo, podporujúci dynamické vyhľadávanie. Vyhľadané kontakty je možné priamo kontaktovať cez vstavanú aplikáciu na volanie, pridať medzi vlastné kontakty, alebo zobraziť na mape. Ukážka zobrazenia mapy je na obrázku 12 vpravo.



Obr. 12: VŠB Telefonní Seznam - ukážka užívateľského rozhrania [7]

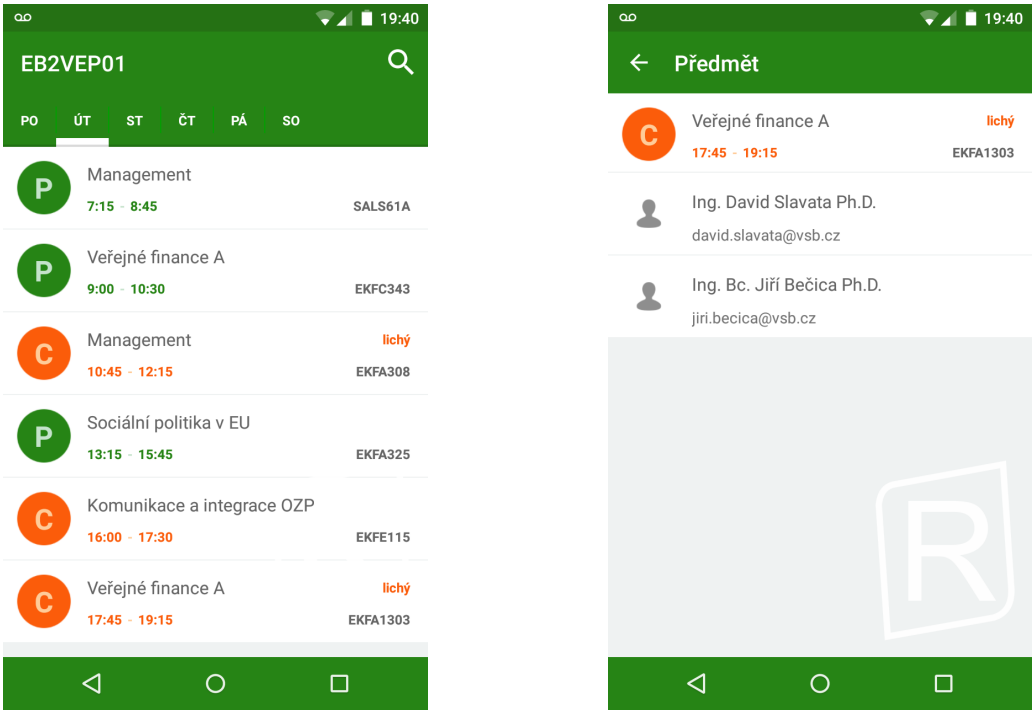
2.2.3 Rozvrh VŠB-TUO

Mobilná platforma: Android (verzia 2.3 a vyššia)
Popisovaná verzia aplikácie: 3.0.1 (zo dňa 10.2.2015)

Dostupné funkcie (spracované podľa [8]):

- Rozvrhy pre študijné skupiny
- Detail rozvrhovej aktivity

Túto aplikáciu som nemohol otestovať v plnej miere, nakoľko ihneď po spustení bola okamžite ukončená a systém zobrazil hlásenie, že aplikácia prestala pracovať. Aplikáciu som testoval na zariadeniach s verziou systému Android 5.1.1, 7.1 a 8.1, no ani na jednom sa nespustila. Z priloženého popisu na *Google Play Store* som však zistil, že aplikácia poskytovala len rozvrhy pre študijné skupiny. Individuálne rozvrhy študentov aplikácia neposkytovala. Na obrázku 13 vľavo je ukážka, ako aplikácia zobrazuje rozvrh pre konkrétnu rozvrhovú skupinu pre konkrétny deň. Obrázok 13 vpravo zobrazuje detail predmetu.



Obr. 13: Rozvrh VŠB-TUO - ukážka užívateľského rozhrania [8]

2.2.4 VŠB Navi

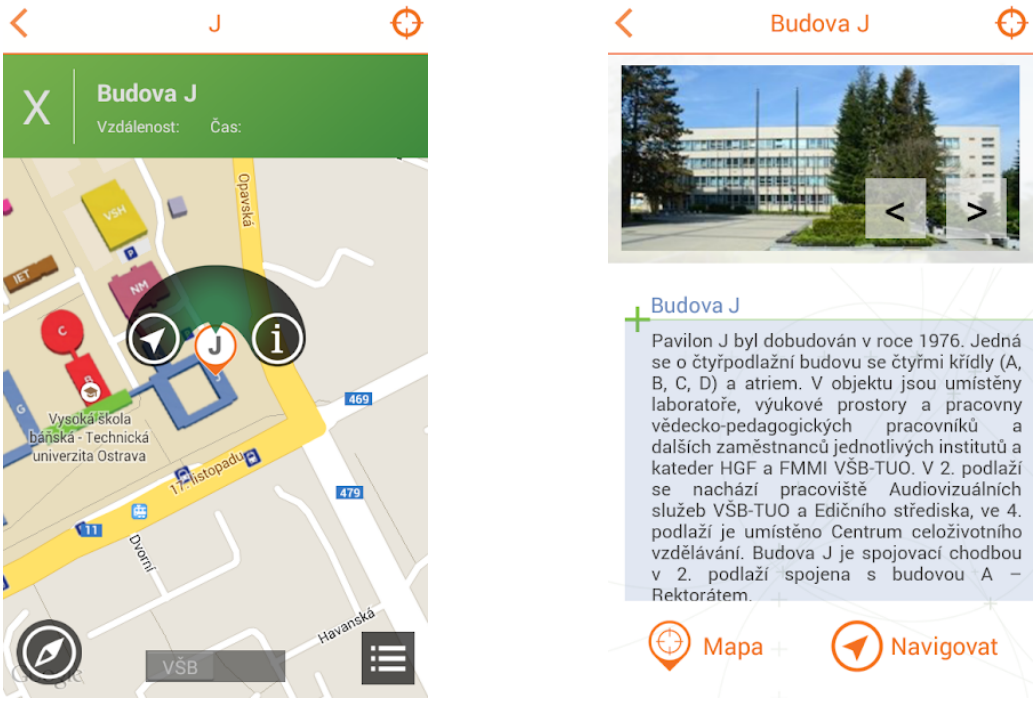
Mobilná platforma: Android (verzia 4.0 a vyššia), iOS (verzia 7.0 a vyššia)
Popisované verzie aplikácií: Android - 2.1 (zo dňa 16.1.2015), iOS - 4.0 (zo dňa 17.12.2015)
Dostupné funkcie (spracované podľa [9]):

- Informácie o bodoch záujmu v areáloch VŠB-TUO spolu s fotografiami
- Navigácia do konkrétnych bodov

- Univerzitné novinky
- Vyhľadávanie kontaktov v adresári
- Mapa areálov s vyznačenými budovami (ukážka je na obrázku 14 vľavo)

Na platforme Android som s aplikáciou zaznamenal rovnaký problém ako v prípade aplikácie *Rozvrh VŠB-TUO*. Aplikácia po úvodnom stiahnutí požadovaných údajov prestala pracovať a následne ju už nebolo možné spustiť. Na operačnom systéme iOS sa však spustila.

Užívateľské rozhranie je veľmi neprehľadné a chaotické. Aplikácia poskytuje relevantné údaje o miestach v areáloch a univerzitných novinkách. V detaile konkrétneho miesta (na obrázku 14 vpravo) je možné spustiť priamu navigáciu. Miesta na mape sa dajú filtrovať podľa kategórii *relax* alebo *jedlo*. Kontakty je možné vyhľadávať cez veľmi jednoduchý formulár, obsahujúci len položky meno, priezvisko a osobné číslo. Po zadaní hľadaného výrazu a spustení vyhľadávania sa však aplikácia po krátkej chvíli ukončí.



Obr. 14: VŠB Navi - ukážka užívateľského rozhrania [9]

2.3 Zhrnutie prehľadu

Stav aplikácii iných univerzít je na dobrej úrovni. Niektoré aplikácie poskytujú základnú, iné rozšírenú funkcionalitu, no v konečnom dôsledku sú všetky aplikácie pre študentov užitočné. Väčšina aplikácii je určená pre platformu Android, čo je škoda, nakoľko aj platforma iOS je medzi študentami značne rozšírená. Spomedzi spomínaných aplikácii by som vyzdvihol užívateľské rozhranie *Uplikace* a *Virtual FIIT*, ktoré pôsobilo najpríjemnejšie. Spracovanie rozvrhu

bolo najlepšie v prípade *Uplikace*. Najčastejšie implementované funkcionality v aplikáciach iných univerzít sú:

- rozvrh
- prihlasovanie sa na termíny skúšok
- prehľad aktuálnych študijných výsledkov

Spomedzi existujúcich aplikácii určených pre študentov VŠB-TUO som zistil, že jediná, plne funkčná a užívateľsky prívetivá aplikácia je *VŠB Telefonní Seznam*. Jedinou jej nevýhodou je absencia prehľadu bodov záujmu v areáloch a informácii o nich. Aplikácia *Průvodce VŠB-TU Ostrava* túto funkcionalitu poskytuje, avšak nemá vyhľadávanie kontaktov a má veľa nadbytočnej funkcionality. *VŠB Navi* je na platforme Android absolútne nepoužiteľná a na iOS je nefunkčné vyhľadávanie kontaktov. Vzhľadom k dátumu poslednej aktualizácie, považujem podporu pre aplikáciu *Rozvrh VŠB-TUO* za ukončenú a samotnú aplikáciu ako nepoužiteľnú v plnej miere, nakoľko ju nie je možné spustiť.

3 Súčasné služby poskytované CIT pre účely orientácie

V tejto časti popíšem existujúce možnosti, ako sa môžu študenti orientovať v jednotlivých areáloch VŠB-TUO pomocou poskytovaných služieb CIT. Jednou z ďalších možností sú tiež 3D panoramatické prehliadky vybraných miestností určitých budov[7]. Túto službu však nezastrešuje CIT, preto v nasledujúcom prehľade nebude zahrnutá.

3.1 Mapy areálov

Na oficiálnej web stránke univerzity sú dostupné mapy areálov VŠB-TUO. Na mape sú farebne vyznačené dôležité budovy spolu s označením. K mape je tiež dostupná legenda, ktorá vymenúva dôležité miestnosti v danej budove. Zobrazenie tiež vyznačuje parkovacie miesta a blízke zastávky mestskej hromadnej dopravy. Zobrazenie mapy je na obrázku 15. Rovnaké mapy spolu s legendou sú dostupné aj ako navigačné tabule rozmiestnené v jednotlivých areáloch.

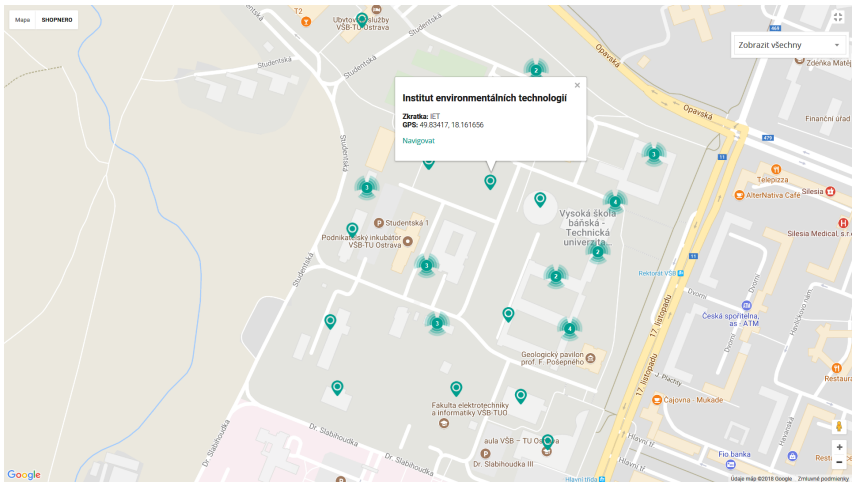
Obr. 15: Ukážka mapy areálu VŠB-TUO v Porube na web stránke univerzity[11]

3.2 Google mapy

Pre účely orientácie je na web stránke univerzity tiež možnosť orientácie pomocou známych a spoľahlivých Máp Google. Na mape sú vyznačené dôležité body. Po kliknutí na bod sa zobrazí

[7] `http://vsb.pano3d.cz/`

jeho názov, skratka, GPS súradnice a tiež možnosť priamej navigácie k danému bodu. V prípade voľby priamej navigácie je užívateľ presmerovaný na stránku Google Maps, kde je ako cieľ prednastavený zvolený bod. Pre navigáciu od aktuálnej polohy užívateľa je potrebné povolenie na určovanie polohy zariadenia, na ktorom užívateľ mapu zobrazuje. Na obrázku 16 je snímka obrazovky prehliadača so zobrazením areálu VŠB-TUO v Porube pomocou Máp Google.



Obr. 16: Ukážka mapy areálu VŠB-TUO v Porube cez Mapy Google

4 RESTful API

Táto kapitola práce sa zameriava najskôr na popis RESTful API vo všeobecnosti, jeho znaky a architektúru. Následne bude uvedený popis súčasného stavu RESTful API, pomocou ktorého je možné pristupovať ku vybraným službám CIT.

4.1 RESTful API všeobecne

Škálovateľnosť webu bola riadená určitými kľúčovými obmedzeniami. V roku 1993, sa Roy Fielding so svojimi spolupracovníkmi rozhodli upraviť implementáciu webu pomocou pragmatického prístupu tak, aby boli dodržané pravidlá škálovateľnosti, no zároveň sa web mohol rozširovať. Odvodil nasledovné princípy - podmienky, ktoré popisujú web ako architektonický štýl:

1. **Klient-server**
   - delenie zodpovednosti za dané úlohy
   - každý komponent sa môže vyvíjať samostatne (iný jazyk alebo technológia)
2. **Jednotné rozhranie**
   - použitím jednotného rozhrania zjednodušíme celkovú architektúru systému a tiež zviditeľníme interakciu medzi jednotlivými komponentami
3. **Vrstvený systém**
   - architektúra je zložená z hierarchických vrstiev, ktoré každá má vlastnú implementáciu ukrytú pred ostatnými vrstvami
   - vrstvy medzi sebou komunikujú iba prostredníctvom rozhraní
   - tiež umožňuje vyvažovanie zaťaženia jednotlivých vrstiev
4. **Cache**
   - v záujme zvýšenia efektivity sieťovej komunikácie môžeme dáta obsiahnuté v odpovedi na požiadavku označiť tak aby mohli byť uložené v klientovej cache pamäti
   - v prípade rovnakej požiadavky sú tieto dáta použité a nie je potrebné ich znovu prenášať cez sieť
5. **Bezstavovosť**
   - každá požiadavka od klienta musí obsahovať všetky potrebné informácie potrebné k pochopeniu požiadavky
   - na serveri sa neukladajú informácie medzi jednotlivými požiadavkami od klienta
6. **Kód na požiadanie** (voliteľná podmienka)

- funkcionalita klienta môže byť rozšírená stiahnutím a vykonaním kódu vo forme appletov alebo skriptov
- zníženie rozsahu funkcionality, ktorá musí byť vopred implementovaná v klientovi

Fielding neskôr so svojimi spolupracovníkmi pracovali na zvyšovaní škálovateľnosti webov. To viedlo ku štandardizovaniu novej verzie HTTP protokolu - verzie 1.1 a formalizácii syntaxi URI. V roku 2000 Fielding pomenoval a popísal tento nový architektonický štýl, ktorý spĺňa vyššie uvedené podmienky, vo svojej dizertačnej práci ako *"Representational State Transfer – REST"*.

Z pohľadu architektúry typu klient-server, klientské programy využívajú API na komunikáciu s webovými službami. API všeobecne, je určené k tomu, aby vystavilo súbor dát a funkcii daného systému navonok a tým dovolilo výmenu dát medzi programami. Web API je rozhranie webovej služby, ktoré priamo načúva klientským požiadavkám a odpovedá na ne. Architektonický model REST je často využívaný pri návrhu API moderných webových služieb. Web API podliehajúce modelu REST sa označuje ako REST API. RESTful API je len iným názvom pre REST API. [29]

4.1.1 Metódy práce s dátovým zdrojom

Rozhranie REST je použiteľné pre jednotný a jednoduchý prístup ku vzdialeným dátam. Všetky zdroje, ku ktorým pomocou tohto rozhrania pristupujeme, majú vlastný identifikátor URI. REST obsahuje štyri základné metódy pre prístup ku dátam a ich úprave.
Tieto metódy sú označované aj ako CRUD. [28] V zátvorke je uvedená odpovedajúca metóda HTTP protokolu:

- Create - vytvorenie (POST)
- Retrieve - získanie (GET)
- Update - úprava (PUT)
- Delete - zmazanie (DELETE)

4.2 Súčasný stav API poskytovaného CIT

Pre účely tejto práce mi bol zo strany CIT poskytnutý prístup k službám RESTful API, ktoré umožňujú pristupovať ku kontaktným údajom, informáciám pre orientáciu v areáli a univerzitným novinkám. Ku každej z poskytnutých služieb mi bola dodaná dokumentácia popisujúca prácu s daným API. Formát dát je buď JSON alebo XML. Tento formát je možné určiť na základe hlavičky HTTP žiadosti *Accept*.

4.2.1 Služby nevyžadujúce prihlásenie

Nasledujúce služby nevyžadujú pri používaní súčinnosť užívateľa. Autorizačný token pre prístup ku týmto dátam sa získa na základe autentifikačných údajov, poskytnutých zo strany poskytovateľa služieb, teda CIT.

Kontaktné údaje

Základným dátovým objektom, s ktorým služba pracuje je *vCard*. Ten zároveň obsahuje názov *organizačnej jednotky* a zoznam *typov funkcie*, ktoré daná osoba zastupuje. Typ funkcie je len číselník, ktorý obsahuje jej názov a identifikátor v databáze. Služba môže vracať údaje taktiež vo formáte *xCard* alebo *jCard*, čo znamená XML alebo JSON reprezentáciu *vCard* formátu.

Služba poskytuje vyhľadanie kontaktu na základe organizačnej jednotky, mena, priezviska, osobného čísla (loginu), telefónu, identifikátoru typu funkcie, čísla miestnosti alebo názvu funkcie. Taktiež je možné sa dopytovať do číselníku typov funkcii alebo organizačných jednotiek. Poskytované údaje sú v českom jazyku.

Orientácia v areáli

Služba obsahuje popis *miest* v jednotlivých areáloch, ktoré sú tiež identifikované *typom* a *kategóriou*. Miesta obsahujú mimo iné aj svoje geografické súradnice, názov a popis v českom a anglickom jazyku, obrázky k danému miestu alebo identifikátor nadradeného miesta. Ďalej tu existuje objekt typu *bod*, obsahujúci napríklad geografické súradnice alebo zoznam miest, ktoré danému bodu náležia. Medzi bodmi existuje *prepojenie*.

Informácie a univerzitné novinky

Pre informácie o novinkách poskytuje služba objekty typu *akcia* a *aktualita*. Ku každej akcii a aktualite môže byť priradený *bod záujmu* akcie alebo *miesto*. Pri testovaní služby som však zistil, že akcie a aktuality neobsahujú žiadne dáta. Ďalej služba obsahuje objekt typu *správa*, ktorý:

- je zaradený do určitej *kategórie*
- je označený je určitým *štítkom*
- môže obsahovať *prílohy* a *obrázky*, ktoré je možné stiahnuť z verejnej URL adresy
- má definovaný *kanál*, teda zdroj
- má *osobu* zodpovednú za daný príspevok

Táto služba už vracia relevantné výsledky a správy sú delené do štyroch kategórii, a to novinky, nadchádzajúce udalosti, ponuka práce a úradné dokumenty. Počas testovania som však zistil, že v kategórii úradné dokumenty nie sú žiadne dáta.

Študijné záležitosti

Služba poskytuje číselník akademických rokov, informácie o jednotlivých semestroch a ich typy (letný alebo zimný). Pri semestroch sa jedná o údaje napríklad počet týždňov výuky, počet týždňov skúškového obdobia alebo názov akademického roku. Služba taktiež zahŕňa formát rozvrhového okna. To je definované svojim identifikátorom, poradím, počiatočným a koncovým časom.

4.2.2 Služby vyžadujúce prihlásenie

Pretože nasledujúce služby pristupujú k osobným údajom, musí sa užívateľ autorizovať svojim osobným číslom a heslom.

Rozvrh

Základným dátovým objektom je *ConcreteActivity*, teda rozvrhová aktivita. Ten obsahuje množstvo atribútov, medzi ktoré patrí napríklad názov predmetu, identifikátor semestru, počet výukových jednotiek nasledujúcich po sebe, čas začiatku a konca výuky, identifikátor dňa v týždni alebo zoznam miestností a vyučujúcich. Každá rozvrhová aktivita je ešte určená objektom *WeekActivity*, ktorá určuje dátum výskytu aktivity.

Informácie o štúdiu

Služba obsahuje jediný dátový objekt s názvom *StudyRelation* reprezentujúci vzťah študenta s univerzitou. Údaje, ktoré objekt nesie sú napríklad začiatok a koniec študijného pomeru, názov fakulty, ročník, názov študijného programu, názov študijného odboru, názov študijnej špecializácie, meno, priezvisko, tituly, osobné číslo alebo školský email.

4.2.3 Navrhované rozšírenia alebo zlepšenia

Považujem za užitočné, aby študenti mohli mať v aplikácii aj svoj vlastný rozvrh a nastaviť si upozornenie na blížiacu sa výuku. Tiež by bolo užitočné aby aplikácia poskytovala informácie o práve prebiehajúcom semestri alebo štúdiu užívateľa. Vtedajší stav API však túto funkcionalitu neponúkal, preto som dal návrh do CIT na pridanie daných funkcii. Mojej žiadosti CIT vyhovelo a službu dodatočne dopracovali. Popis doplnených služieb je zahrnutý v sekcii 4.2.

5 Návrh aplikácie

V nasledujúcej kapitole popíšem návrh aplikácie. Inšpiráciou pre rozširujúce funkcie, ako aj užívateľské rozhranie mi budú aplikácie rozoberané v kapitole 2.

V procese vývoja softvéru je analýza problému a návrh riešenia kľúčovým krokom. Umožňuje rozložiť komplexný systém na menšie celky a popísať systém z rôznych hľadísk.

5.1 Vízia

Vízia predstavuje základnú predstavu o výslednom systéme, zľahka popisuje jej prvoradú funkcionalitu a aktérov pracujúcich so systémom. Je napísaná tak, aby bola pochopiteľná ako zákazníkom (laikom), tak aj riešiteľom (programátorom). Vízia tiež popisuje, aké ciele sa nový systém snaží dosiahnuť, teda akú pridanú hodnotu prinesie.

Chceme vytvoriť mobilnú aplikáciu určenú pre minimálne dve mobilné platformy. S aplikáciou budú pracovať študenti VŠB-TU Ostrava, ktorým bolo pridelené osobné číslo (login) a heslo. S obmedzenou funkcionalitou bude použiteľná aj pre bežných návštevníkov univerzity. Aplikácia bude poskytovať jednoduchý prístup ku kontaktným informáciám zamestnancov univerzity, univerzitným novinkám, informáciám pre navigáciu v rámci jednotlivých areálov, rozvrhu študenta a informáciám o štúdiu študenta.

Aplikácia uľahčí hlavne novým, ale aj starším študentom rýchlejšie sa orientovať medzi budovami a miestnosťami. Študenti tiež môžu byť častejšie informovaní o univerzitných novinkách, alebo pohodlnejšie kontaktovať svojich vyučujúcich. Študenti si budú môcť nastaviť upozornenia na blížiacu sa výuku, aby predišli nechcenému zmeškaniu výuky. Hlavnou výhodou je, že aplikáciu budú mať takmer neustále na dosah ruky, takže sa rýchlejšie dostanú k potrebným informáciám.

5.2 Špecifikácia požiadaviek

Jasnou špecifikáciou požiadaviek konkrétnejšie definujeme, čo od výsledného softvéru očakávame a ako budú tieto požiadavky plnené. Jedná sa v podstate o rozšírenú víziu so zameraním na požiadavky.

Nasledujúce požiadavky vychádzajú zo základných požiadaviek stanovených zadaním tejto práce a tiež z prieskumu existujúcich aplikácii z kapitoly 2. Zahŕňajú funkcionalitu, ktorú by štandardne mala poskytovať dobrá univerzitná aplikácia, aby bola prakticky použiteľná pre študentov.

Navigácia

Základnou požiadavkou je poskytnutie funkcionality pre orientáciu v areáli VŠB-TUO. To bude riešené formou mapy, kde budú zaznačené dôležité strategické body, najmä budovy, ktorých polohu poskytuje CIT vo svojom API. Po kliknutí na dané miesto užívateľ môže spustiť navigáciu

k tomuto miestu. Na mape bude taktiež možné zobraziť polohu miestnosti, ktorá je spojená s kontaktnými informáciami v API telefónneho zoznamu. Aplikácia tiež bude poskytovať detailné informácie o daných miestach.

Kontakty

Aplikácia bude obsahovať jednoduchý formulár na vyhľadanie ľudí na VŠB-TUO. Táto funkcia bude vyžadovať pripojenie k internetu. Pre vyhľadané kontakty si užívateľ môže zobraziť kompletné informácie, ktoré poskytuje CIT. Vybrané kontakty si užívateľ môže uložiť do obľúbených a bude ich mať dostupné aj bez pripojenia. Kontakty, ktoré obsahujú aj telefónne čísla bude možné priamo zavolať alebo odoslať SMS cez vstavané aplikácie na to určené. Ak kontakt obsahuje mailovú adresu, bude možné na ňu odoslať mail pomocou vstavaného mailového klienta. Pri kontaktoch, ktoré obsahujú aj označenie miestnosti a jej polohu, bude poskytnuté prepojenie na mapu a následná navigácia do danej miestnosti.

Aktuality

Študenti si budú môcť zobraziť univerzitné aktuality priamo v aplikácii, bez potreby otvárania internetového prehliadača. Budú vypísané pod sebou chronologicky s reprezentatívnym obrázkom a nadpisom. Po kliknutí sa zobrazia detailnejšie informácie. Aktuality budú rozdelené podľa kategórie.

Rozvrh

Študent po prihlásení do aplikácie so svojim univerzitným účtom bude môcť zobraziť svoj aktuálny rozvrh. Rozvrhové okno bude obsahovať základné informácie o výuke a po kliknutí bude možné zobraziť bližšie informácie o predmete, kontaktné údaje na vyučujúceho alebo navigáciu do danej miestnosti. Ďalej bude dostupná možnosť nastaviť si upozornenie na určitú dobu pred začatím výuky, aby mal študent dosť času sa dostaviť do danej miestnosti. Spôsob upozornenia ako aj čas pred začatím bude možné nastaviť individuálne pre každé rozvrhové okno.

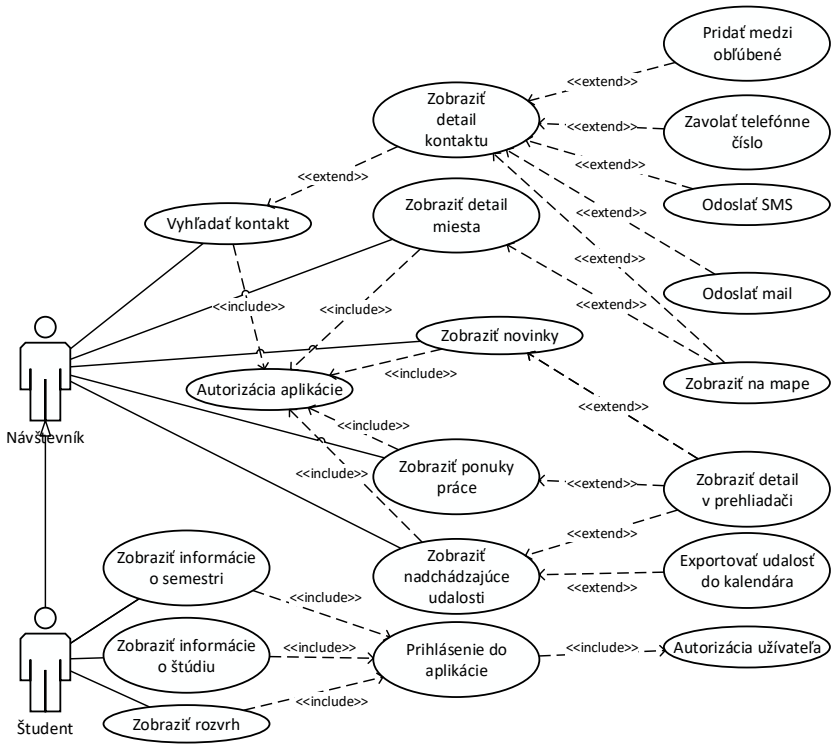
Informácie o štúdiu

Aplikácia poskytne prihlásenému užívateľovi informácie o jeho aktuálne prebiehajúcom štúdiu. Bude sa jednať najmä o názov fakulty, názov študijného programu a odboru, osobné číslo, školský email a podobne.

5.2.1 Use Case diagram

Diagram, ktorý zobrazuje systém z pohľadu aktérov spolupracujúcich so systémom spolu s činnosťami, ktoré v systéme vykonávajú. Ide teda o určenie, kto môže v systéme vykonávať ktoré akcie a v akom rozsahu a tiež aké iné úkony tieto akcie zahŕňajú.

Keďže so systémom môže pracovať buď prihlásený študent, alebo návštevník univerzity, *Use Case* diagram na obrázku 17 obsahuje celkovo dvoch aktérov. Prihlásený užívateľ môže v systéme vykonávať rovnaké činnosti ako neprihlásený.



Obr. 17: Use Case diagram

5.3 Návrh užívateľského rozhrania

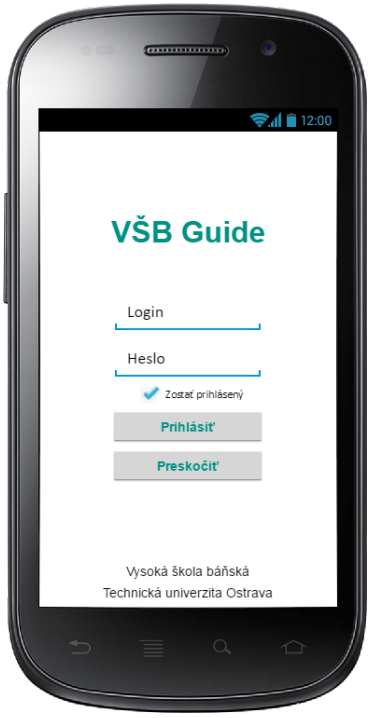
Prvotné návrhy užívateľského rozhrania dávajú aj nezainteresovanej osobe jasnejšiu predstavu o finálnom produkte. Mali by byť preto neoddeliteľnou súčasťou každého návrhu systému. Je to spôsob prezentácie softvérového diela navonok, preto by jeho dizajn mal byť starostlivo zvážený a viackrát prekonzultovaný oboma zúčastnenými stranami - zadávateľom aj riešiteľom.

Moderné poznatky o návrhu užívateľského rozhrania hovoria, že by malo pôsobiť pokojne a nemalo by užívateľa rušiť alebo znepokojovať. Grafické prvky na jednotlivých obrazovkách by mali byť logicky zoskupené a uložené, mala by byť dodržaná pravidelnosť a prehľadnosť. Nemalo by dochádzať ku miešaniu väčšieho množstva rôznych typov a farieb písma. Ďalej podľa môjho názoru užívateľ musí byť neustále informovaný o tom, čo sa v aplikácii aktuálne odohráva. Tým myslím napríklad aby počas načítavania obsahu bol zobrazený indikátor aktivity, ktorý dáva najavo, že aplikácia stále reaguje a čaká na dokončenie sekcie programu v pozadí.

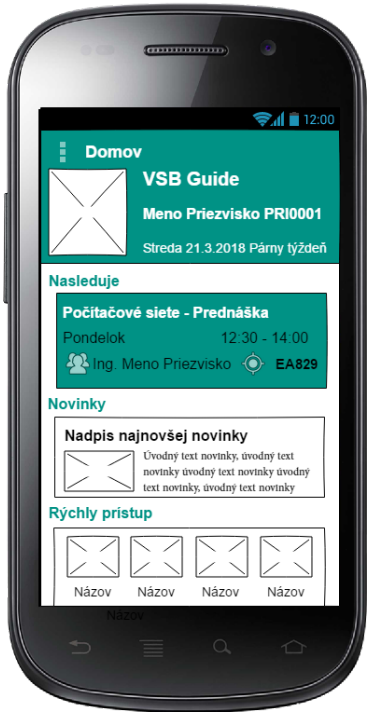
V neposlednom rade, keďže sa jedná o mobilnú aplikáciu, ktorá bude spúšťaná na obrazovkách rôznych veľkostí a rozlíšení, je nutné zabezpečiť správne zobrazenie na akomkoľvek zariadení. Jedná sa nielen o rozloženie prvkov, ktoré sa na väčších obrazovkách usporiadajú rovnako

ako na malých, ale aj o použité statické obrázky, ktorých kvalita by nemala byť degradovaná s rastúcou uhlopriečkou.

Na obrázku 18 je zobrazený návrh prihlasovacej obrazovky aplikácie. Tá obsahuje len jednoduchý formulár pre zadanie prihlasovacích údajov a možnosť preskočenia prihlásenia v prípade užívateľa bez prihlasovacích údajov. Po tejto obrazovke je užívateľ presmerovaný na domovskú obrazovku zobrazenú na obrázku 19. Jej obsah tvorí banner s informáciami o užívateľovi, okno s nasledujúcou výukou, okno s najnovšou novinkou a sekcia s rýchlym prístupom. Pre vyhľadávanie kontaktov slúži formulár na obrázku 20, odkiaľ je po kliknutí na vyhľadaný kontakt užívateľ presmerovaný na obrazovku s detailom kontaktu na obrázku 21.



Obr. 18: Návrh prihlasovacej obrazovky



Obr. 19: Návrh domovskej obrazovky

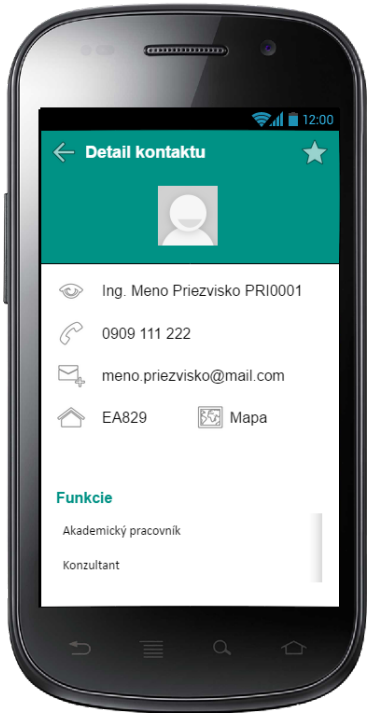
5.4 Technická špecifikácia

Táto sekcia sa venuje technickému pohľadu na navrhovaný softvér. Technická špecifikácia definuje, mimo iné, aj cieľové zariadenia a platformy, pre ktoré bude výsledný softvér určený. Tiež popisuje, pomocou akej technológie budú jednotlivé komponenty systému implementované a môže určiť aj implementačné prostredie použité pri vývoji.

5.4.1 Hardvérová a softvérová špecifikácia

Aplikácia bude riešená ako mobilný klient pre viac platforiem. Ako môžeme vidieť na grafe na obrázku 1 v kapitole 1.1.1, najrozšírenejšími platformami, pre ktoré je ešte vhodné zaoberať

Obr. 20: Návrh obrazovky hľadania kontaktov



Obr. 21: Návrh obrazovky detailu kontaktu

sa vývojom nových aplikácii, sú Android a iOS. Preto tieto dve platformy budú cieľové aj pre túto aplikáciu. V prípade OS Android, na základe grafu na obrázku 2 v kapitole 1.1.2, môžeme vidieť, že najstaršia verzia, ktorú má zmysel ešte podporovať je verzia 4.4. Čo sa týka operačného systému iOS, na základe grafu na obrázku 3 v kapitole 1.1.2, som zvolil za najstaršiu podporovanú verziu iOS 9.3. Tento výber zaistí kompatibilitu s čo najširším počtom aktívnych zariadení a zároveň umožní využívať funkcionalitu moderných verzii operačných systémov.

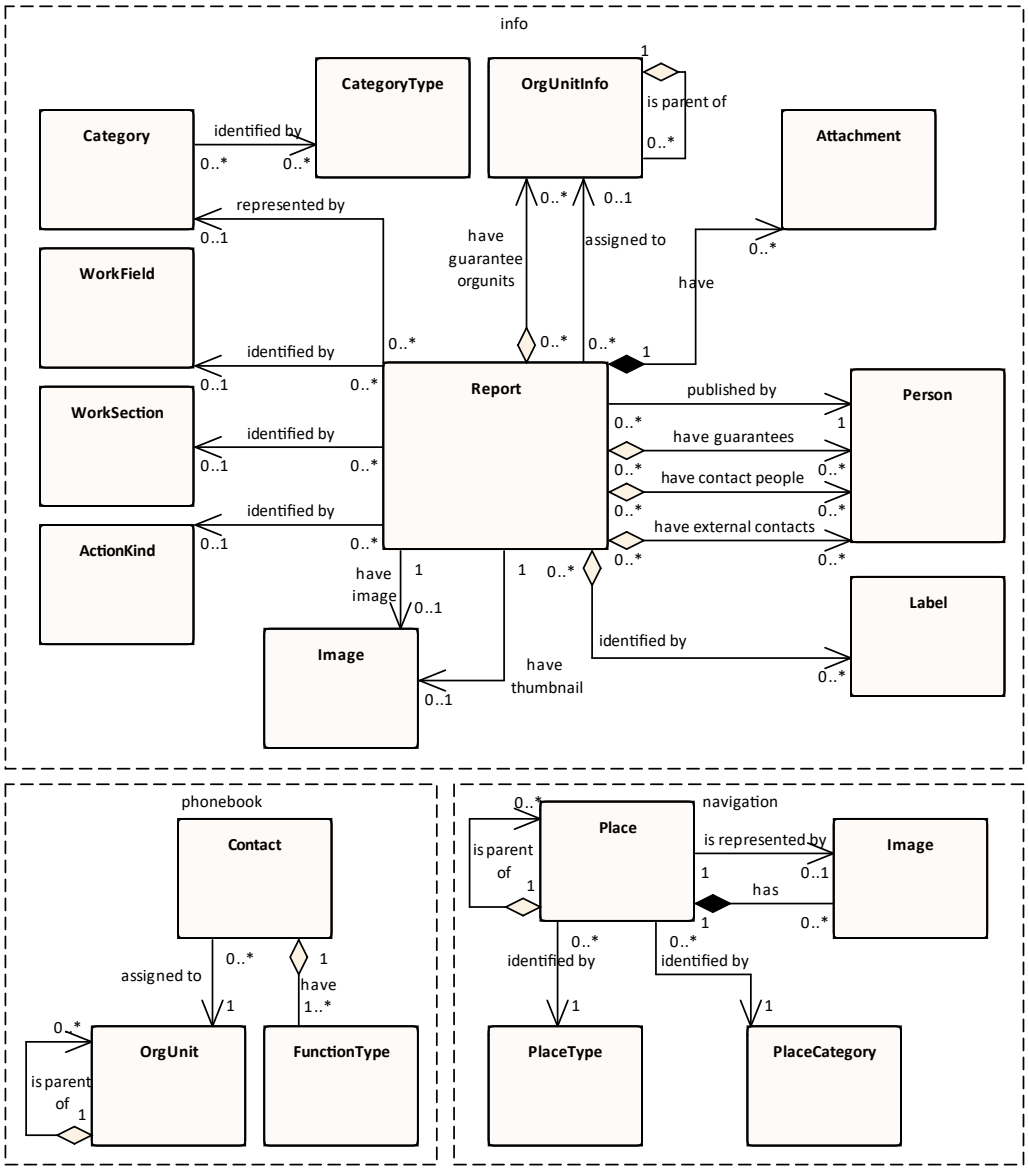
5.4.2 Použité technológie a implementačné prostredie

Pre vývoj tejto aplikácie som zvolil technológiu *Xamarin*, pretože je písaná v jazyku C#, ktorý mi je bližší, než HTML5 využívaný v *Apache Cordova*. Ďalším dôvodom tohto výberu je dostupnosť rozsiahlej podpornej komunity, ktorá časom vznikla okolo tejto technológie. Pomocou *Xamarin* je možné vyvíjať aplikácie pre OS Android od verzie 4.0.3 a pre iOS od verzie 8, čo je vyhovujúce zadaným softvérovým požiadavkom. [13]

Ako implementačné prostredie bude použité *Microsoft Visual Studio 2017*, verzia *Enterprise*. Jedná sa o robustný nástroj využívaný v mnohých softvérových firmách profesionálnymi vývojármi a poskytuje širokú paletu rôznych doplnkov pre efektívny vývoj a testovanie. Ďalším dôvodom, voľby práve tohto nástroja je fakt, že v dobe písania tejto práce je práve *Microsoft Visual Studio* oficiálnym vývojovým prostredím pre technológiu *Xamarin*. Nahradil tak dovtedy podporovaný nástroj *Xamarin Studio*, ktorého verzia 6.3 bola posledná a ďalej sa už nevyvíjal. Pre správu verzii kódu bude použitý *Visual Studio Team Services*.

5.5 Návrh doménového modelu

Základ pre návrh domény tvoria objekty, obsiahnuté v API, s ktorým bude aplikácia komunikovať. Medzi nimi existujú jednoduché vzťahy maximálne typu 1:N. Pretože všetky triedy, ktoré som implementoval sú s anglickými názvami, nasledujúce diagramy obsahujú triedy a ich metódy tiež s anglickými názvami.



Obr. 22: Doménový model

5.5.1 Prvý model domény

Doménový model je forma triedneho diagramu, ktorý predstavuje základný návrh entít systému a vzťahy medzi nimi. Nie je závislý na platforme, preto pri atribútoch tried nie sú uvedené ich

dátové typy. [21] U tried sa zároveň nemusia neuvádzať metódy a obsahujú len tie najdôležitejšie atribúty. Diagram obsahuje iba tzv. biznis triedy, ktoré modelujú problémovú oblasť. [22]

Služby, ktoré poskytuje API sú členené do troch rozsahov:

- *infoservice* - univerzitné novinky
- *navigation* - navigácia a udalosti
- *phonebook* - adresár kontaktov

Pre každú službu je potrebné vytvoriť objekt, do ktorého sa uložia získané dáta. Objekty môžu obsahovať referencie na iné objekty, prípadne len ich jednoznačné identifikátory. Pre prácu s databázou sa použije návrhový vzor *Table Data Gateway*[8] a pre biznis logiku *Transaction Script*[9]. Preto pre každú triedu predstavujúcu dátový objekt sa vytvorí príslušná trieda `Transactional` a jedna spoločná generická trieda `Gateway`. Ďalej budú potrebné triedy zaisťujúce autorizáciu aplikácie, synchronizáciu databázy s API a prácu s aplikačným úložiskom. Pre zachovanie prehľadnosti diagramu doménového modelu na obrázku 22 sú zobrazené len doménové objekty, ktoré boli využité pri tejto implementácii a sú súčasťou poskytovaných služieb API. Ostatné potrebné triedy sú zahrnuté v statickom diagrame tried v sekcii 5.5.2.

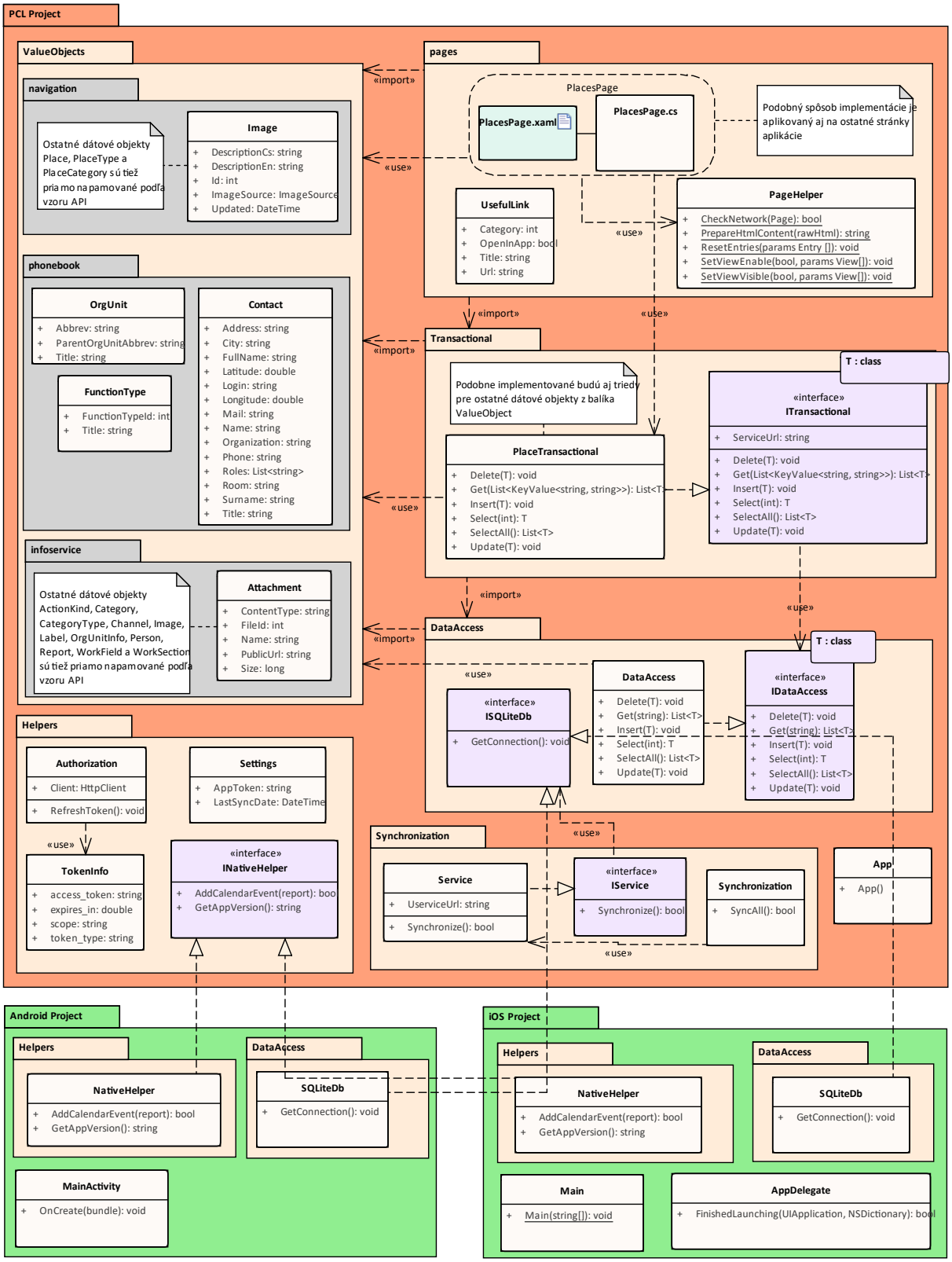
5.5.2 Statický diagram tried

Diagram tried popisuje statickú štruktúru modelovaného systému. Zobrazuje triedy, ich stav (atribúty) a správanie (metódy) a tiež vzťahy medzi triedami a ich kardinalitu. [19] U atribútov a metód sa tiež definujú ich modifikátory prístupu. Môže tiež zobrazovať systém z pohľadu balíkov a menných priestorov. Musí zodpovedať zadaným funkčným požiadavkom, scenárom z *Use Case* diagramu ale aj ostatným diagramom a návrhom. Neskôr môže byť nástrojom pre dokumentáciu. Je závislý na platforme, teda je zobrazený s ohľadom na použitý programovací jazyk. [20]

Celý projekt bude rozdelený do troch triednych knižníc. Jedna bude tvoriť spoločný kód, čo znamená, že bude obsahovať triedy pre dátové objekty, triedy pre doménovú logiku, triedy pre prístup k dátam, triedy pre synchronizáciu a autorizáciu a ostatné podporné triedy. Ďalšie dve triedne knižnice budú predstavovať špecifický kód pre konkrétne mobilné platformy - Android alebo iOS. Budú využívať knižnicu tried spoločného kódu a obsahovať budú príslušné zdrojové súbory (napríklad obrázky), spúšťaciu triedu a iný podporný kód špecifický pre danú platformu. Na obrázku 23 je zobrazený statický diagram tried výslednej aplikácie. Vzhľadom k rozsiahlosti systému a pre zachovanie prehľadnosti diagramu sú triedy s podobnou implementáciou vynechané a sú zobrazené len ich ukážkové implementácie.

[8] `https://martinfowler.com/eaaCatalog/tableDataGateway.html`

[9] `https://martinfowler.com/eaaCatalog/transactionScript.html`



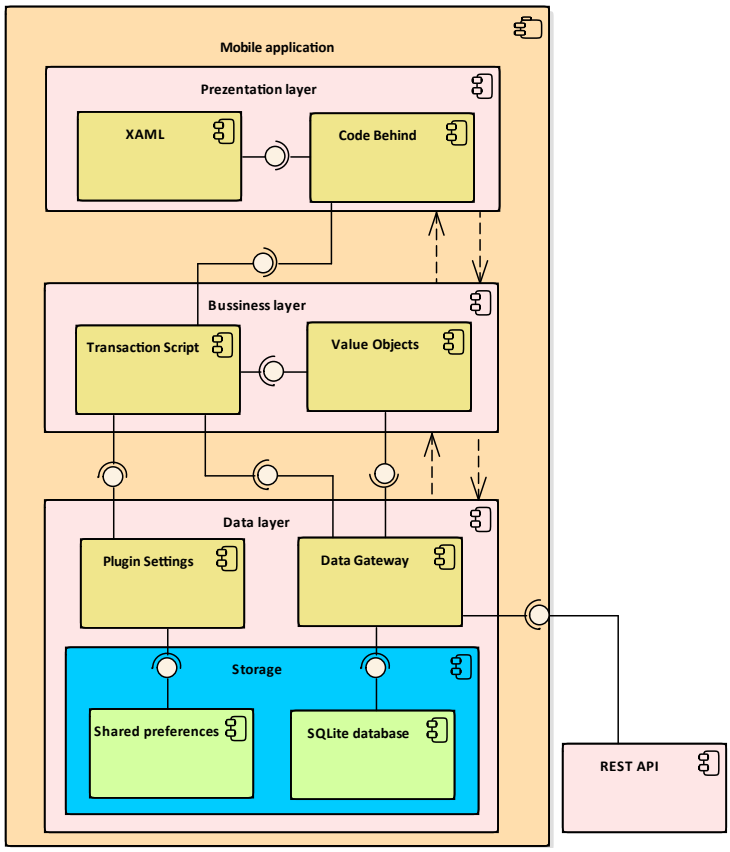
Obr. 23: Statický diagram tried

5.6 Popis architektúry softvéru

Softvér bude riešený ako mobilná aplikácia, ktorá komunikuje so vzdialeným RESTful API.

5.6.1 Diagram komponent

Diagram komponent opisuje prepojenie a komunikáciu komponent tvoriacich systém. Medzi komponentami môže byť vzťah typu závislosť alebo komunikácia. Komponenta je vnímaná ako zapuzdrená modulárna jednotka systému, ktorá môžu byť samostatne zameniteľná, distribuovaná alebo aktualizovaná. Na diagrame sú zobrazené jednotlivé komponenty s poskytovanými a vyžadovanými rozhraniami. [24]



Obr. 24: Diagram komponent

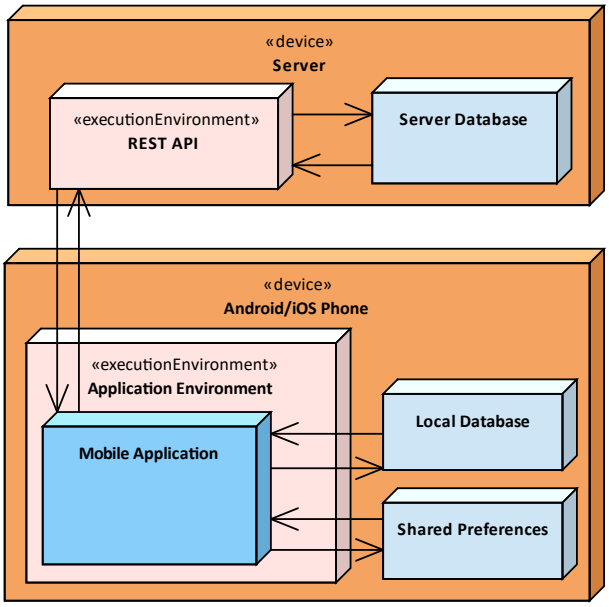
Ako môžeme vidieť na obrázku 24, aplikácia bude riešená ako troj-vrstvová architektúra. Prezentačná vrstva bude riešená pomocou technológie *Xamarin.Forms*, takže každá stránka aplikácie bude predstavovaná jedným *.xaml* a jedným *.cs* súborom. Prezentačnej vrstve bude zabezpečovať biznis logiku ďalšia vrstva, kde je implementovaný návrhový vzor *Transaction Script*. Vrstva biznis logiky tiež zahŕňa všetky dátové objekty, s ktorými pracujú všetky tri vrstvy. Poslednou vrstvou je vrstva prístupu ku dátam, ktorá je čiastočne implementovaná pomocou vzoru *Table Data Gateway*. Táto vrstva pristupuje ku dátam v lokálnej databáze, v úložisku

aplikačných dát a tiež zabezpečuje komunikáciu so vzdialeným RESTful API. Dáta z API budú len získavané, nebude dochádzať ku ich zmene a budú sa ukladať do lokálnej databázy pre použitie bez pripojenia.

5.6.2 Diagram nasadenia

Popis fyzickej architektúry systému znázorňuje diagram nasadenia. Diagram zobrazuje rozloženie jednotlivých softvérových komponent na fyzických (hardvérových) uzloch a ich vzájomnú komunikáciu. Uzly môžu byť typu *«device»* predstavujúci fyzické zariadenie alebo *«execution environment»* predstavujúci prostredie pre beh softvéru. [23]

RESTful API, ktoré využíva aplikácia bude spustené na serveri CIT a bude poskytovať dáta uložené v databáze na serveri. Komunikácia bude prebiehať s mobilnou aplikáciou, ktorá bude spúšťaná na mobilných telefónoch s OS Android alebo iOS. Aplikácia bude tiež komunikovať s lokálnou databázou zariadenia a úložiskom pre aplikačné dáta. Diagram nasadenia je zobrazený na obrázku 25.



Obr. 25: Fyzická architektúra systému

6 Implementácia

V nasledujúcej časti popíšem implementáciu vybraných funkcionalít výslednej aplikácie. Vychádzať pri nej budem z vykonaného návrhu v predošlej kapitole. Popíšem tiež knižnice, ktoré boli do implementácie zahrnuté pre zaistenie požadovanej funkcionality.

6.1 Autorizácia aplikácie

Pre prístup ku službám API, ktoré nevyžadujú súčinnosť užívateľa a nepristupujú ku osobným údajom je nutné použiť autorizačný protokol *Client Credential Flow*. Pomocou neho aplikácia získa prístupový token s ktorým autorizuje každú žiadosť odoslanú na API. Proces autorizácie zabezpečuje statická trieda `Authorization` a jej metóda `RefreshTokenAsync():Task<bool>`. Tá je vyvolaná pred každým zavolaním služby, aby sa zaistilo, že token bude v čase volania služby platný. Metóda najskôr zistí, či tokenu uloženému v aplikačných dátach ešte nevypršala platnosť. Ak áno, vyžiada si nový a ten potom spätne uloží pomocou triedy `Settings`. Získanie nového tokenu zabezpečuje asynchrónna metóda `GetTokenAsync():Task<TokenInfo>` zobrazená na výpise 1. Token má obvykle platnosť 86400 sekúnd, čo predstavuje 24 hodín.

Komunikáciu s API pri získavaní tokenu zabezpečuje inštancia triedy `HttpClient`. Tá implementuje POST metódu ako asynchrónnu, pretože je vykonanie môže určitú chvíľu trvať, v závislosti od dostupnosti služby alebo objemu prenášaných dát. Pretože sa jedná o metódu náchylnú na vyvolanie výnimiek, je jej volanie zahrnuté v `try-catch` bloku. Pred volaním je potrebné vyplniť parametre žiadosti. Zadávajú sa vo formáte *application/x-www-form-urlencoded* a v tejto implementácii sú obsiahnuté v dátovej štruktúre typu `Dictionary` a sú zobrazené na výpise 1. Jednotlivé položky majú nasledovný význam:

- `grant_type` - typ prístupu
- `client_id` - jednoznačný identifikátor aplikácie (pridelilo CIT)
- `client_secret` - prístupové heslo aplikácie (pridelilo CIT)
- `scope` - služby, ku ktorým chceme získať prístup, oddelené medzerou

Služba vráti odpoveď vo formáte JSON, ktorá obsahuje:

- `access_token` - prístupový token pre volanie API
- `token_type` - typ tokenu
- `expires_in` - platnosť tokenu v sekundách
- `scope` - služby, pre ktoré je token platný

```
public static Dictionary<string,string> Values = new Dictionary<string,string>{
    {"grant_type", "client_credentials"},
    {"client_id", /*identifikátor_aplikácie*/},
    {"client_secret", /*heslo_aplikácie*/},
    {"scope", "phonebook infoservice navigation"}
};

private static async Task<TokenInfo> GetTokenAsync()
{
    using(HttpClient httpClient = new HttpClient()) {
        var content = new FormUrlEncodedContent(Values);
        try {
            var response = await httpClient.PostAsync(Settings.AuthUrl,content);
            string responseJson = await response.Content.ReadAsStringAsync();
            TokenInfo token = JsonConvert.DeserializeObject<TokenInfo>(
                responseJson);
            return token;
        }
        catch (Exception exception) {
            Crashes.TrackError(exception);
            return null;
        }
    }
}
```

Výpis 1: Implementácia metódy pre získanie prístupového tokenu

6.2 Synchronizácia

Základný typ, predstavujúci službu, ktorá sa má synchronizovať s databázou, je typ `Service` implementujúci rozhranie `IService`. Toto rozhranie predpisuje jedinú metódu `Synchronize() :Task<bool>`, ktorá sa stará o zosynchronizovanie dát získaných z API s lokálnou databázou. Trieda `Service` tiež obsahuje URL adresu služby, z ktorej sa získajú potrebné dáta. Adresa je zadávaná pri vytváraní novej služby pomocou konštruktora.

Metóda `Synchronize():Task<bool>`, zobrazená na výpise 2, najskôr získa všetky dáta pre danú službu z API. Následne ich vloží do databázy. Pre získavanie a ukladanie dát je využitá inštancia triedy `DataAccess<T>` z dátovej vrstvy. Pre vloženie viacerých dát do databázy naraz sa využíva metóda `UpdateOrInsertAll(List<T>):void`, ktorá je implementovaná tak, aby sa snažila najskôr dáta do tabuľky vložiť. Ak sa dáta v tabuľke už nachádzajú, tak ich aktualizuje.

```
public async Task<bool> Synchronize() {
    DataAccess<ServiceType> dataAccess = new DataAccess<ServiceType>();
    List<ServiceType> insertedObjects = await dataAccess.GetAll(Transactional.
        AddLangParam(ServiceUrl), MimeType.JSON);
    if (insertedObjects is null || !insertedObjects.Any()) {
        return false;
    }
    dataAccess.UpdateOrInsertAll(insertedObjects);
    return true;
}
```

Výpis 2: Implementácia metódy pre synchronizáciu služby

O celkovú synchronizáciu sa stará trieda `Synchronization`. Obsahuje zoznam inštancii typu `IService`, a pri potrebe synchronizácie je na ne v metóde `SyncAll(bool):Task<bool>` zavolaná spomínaná metóda `Synchronize():Task<bool>`. V prípade, pokusu o vykonanie plánovanej synchronizácie metóda najskôr overí, či je synchronizácia potrebná. V prípade vynútenej synchronizácie (keď ju užívateľ vyvolá manuálne v nastaveniach aplikácie) sa overovanie vynecháva. Po úspešnej synchronizácii sa do úložiska aplikačných dát zaznamená dátum a čas poslednej synchronizácie pomocou triedy `Settings`.

Trieda `Synchronization` tiež implementuje metódu `SyncEnabled():bool`, ktorá vracia údaj, či je možné vykonať synchronizáciu. Volať by sa mala pred zavolaním metódy na zosynchronizovanie databázy. Metóda berie v prvom rade do úvahy, či je dostupné pripojenie k internetu, a v prípade, že má užívateľ povolenú synchronizáciu len pri pripojení cez WiFi aj typ aktuálneho pripojenia. Ďalej trieda obsahuje udalosť `event ProgressBarStepper():IncrementProgressBar`, vyvolanú po každom úspešnom zosynchronizovaní služby. Tá je určená na to, aby sa k nej priradila metóda, ktorá posunie stav ukazovateľa indikujúceho aktuálny stav synchronizácie.

6.3 Databáza

Pre prístup ku databáze je potrebné pracovať s inštanciou triedy `SQLiteConnection`. Keďže každá platforma definuje iné miesto, kde je databáza uložená, je potrebné pre získanie pripojenia k databáze implementovať rozdielny postup pre každú platformu. Preto je v knižnici spoločného kódu definované rozhranie `ISQLiteDb`, ktoré predpisuje jedinú metódu `GetConnection():SQLiteConnection`. Následne sú v knižniciach pre platformovo-špecifický kód vytvorené triedy `SQLiteDb`, ktoré toto rozhranie implementujú. Tieto triedy sú označené atribútom uvedeným na výpise 3. Daný atribút indikuje, že táto trieda poskytuje konkrétnu implementáciu požadovaného rozhrania, čo je v tomto prípade rozhranie `ISQLiteDb`. Atribút je potrebný pre správne zavolanie požadovanej metódy zo spoločného kódu.

```
[assembly: Dependency(typeof(SQLiteDb))]
```

Výpis 3: Atribút, ktorým sú označené triedy `SQLiteDb`

Implementovaná metóda `GetConnection():SQLiteConnection` najskôr vytvorí z cesty ku špecifickému priečinku a názvu databázy jej úplnú cestu. Následne sa vytvorí inštancia triedy `SQLiteConnection` pomocou jej konštruktora, ktorý prijíma ako prvý parameter inštanciu typu `ISQLitePlatform` a druhý cestu k databázovému súboru. Na výpise 4 je zobrazená kompletná implementácia triedy `SQLiteDb` pre platformu Android.

```
public class SQLiteDb : ISQLiteDb {
    private static readonly String databaseName = "VSBGuideSQLite.db3";

    public SQLiteConnection GetConnection() {
        string documentsPath = System.Environment.GetFolderPath(System.
            Environment.SpecialFolder.MyDocuments);
        var path = Path.Combine(documentsPath, databaseName);
        if(!File.Exists(path)) {
            File.Create(path);
        }
        return new SQLiteConnection(new SQLitePlatformAndroidN(), path);
    }
}
```

Výpis 4: Implementácia triedy pre získanie pripojenia k databáze pre platformu Android

Pre prácu s pripojením k databáze je použitý konštrukt `using`, ktorý sa postará o vytvorenie inštancie pripojenia a po skončení práce aj o jeho uzavretie. Pre získanie inštancie pripojenia je potrebné použiť metódu zo statickej triedy `DependencyService`, ktorá zabezpečuje získanie platformovo špecifickej implementácie daného typu. V tomto prípade sa teda jedná o typ `ISQLiteDb`. Ukážka získania a práce s pripojením k databáze sa nachádza na výpise 10.

6.4 Mapy a poloha

Pred použitím máp na platforme iOS bolo potrebné v súbore *Info.plist* pridať niekoľko elementov, ktoré oprávňujú aplikáciu využívať polohu užívateľa neustále alebo len počas používania aplikácie. Ukážka pridaných elementov je na výpise 5, kde je zobrazená časť súboru *Info.plist*. Druhým, a zároveň posledným krokom je inicializácia máp pri spustení aplikácie. Na to slúži metóda `FormsMaps.Init():void` volaná v metóde `FinishedLaunching(UIApplication, NSDictionary):bool` v triede `AppDelegate`.

```
<key>NSLocationAlwaysUsageDescription</key>
    <string>Can we use your location</string>
<key>NSLocationWhenInUseUsageDescription</key>
    <string>We are using your location</string>
```

Výpis 5: XML elementy v súbore *Info.plist* zaisťujúce funkčnosť máp na platforme iOS

V prípade OS Android bolo potrebných viac krokov. Prvým krokom bolo získanie prístupového kľúča ku *Google Maps API v2* pomocou vývojárskej konzoly Google[10]. Následne sa kľúč vložil do konfiguračného súboru *AndroidManifest.xml*. V tomto súbore bolo tiež potrebné špecifikovať potrebné povolenia pre používanie polohy zariadenia. Zoznam povolení spolu s API kľúčom pre mapy zapísané v konfiguračnom súbore, sú zobrazené na výpise 6.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.
    ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="
    <!-- API kľúč -->" />
```

Výpis 6: Časť súboru *AndroidManifest.xml*

Mapy sa v aplikácii využívajú na zobrazenie bodov záujmu, poskytované službou RESTful API v časti *navigation* alebo tiež polohy priradenej pri kontaktných informáciach. Poloha sa zaznačí do mapy vytvoreným inštancie triedy `Pin` z rovnakého menného priestoru ako samotné mapy, teda `Xamarin.Forms.Maps`. Pri inicializácii je stred mapy nastavený na požadované miesto. V prípade zobrazenia so všetkými bodmi záujmu, sa mapa centruje na približný stred oblasti kampusu VŠB-TUO v Porube. Tieto súradnice sú definované ako konštanty v triede `Settings`.

6.5 Užívateľské rozhranie

Spôsob, akým sa aplikácia prezentuje užívateľovi má na starosti prezentačná vrstva. Pre jej implementáciu som zvolil multiplatformový prístup. To znamená, že rozhranie pre každú obrazovku

[10] `https://docs.microsoft.com/en-us/xamarin/android/platform/maps-and-location/maps/obtaining-a-google-maps-api-key?tabs=vswin`

aplikácie je tvorené spoločným *XAML* súborom. Jednotlivé *XAML* elementy sa po kompilácii pretvoria na natívne zobrazovacie a ovládacie prvky danej platformy. Ku každému *.xaml* súboru sa viaže jeden *.cs* súbor, implementujúci správanie prezentačnej vrstvy danej obrazovky.

Pre rýchlejšie vytváranie a ladenie rozloženia prvkov jednotlivých obrazoviek som využíval nástroj *Gorilla Player*[11]. Pomocou neho som zmeny v *.xaml* súbore videl okamžite na viacerých typoch zariadení naraz bez potreby zdĺhavej kompilácie a inštalácie aplikácie.

Domovská obrazovka

Pre uloženie prvkov na domovskej obrazovke je použitý koreňový element `RelativeLayout`, ktorého vnorené elementy sa zobrazujú na základe určitých parametrov buď voči priamo nadradenému elementu, alebo voči ľubovoľnému inému elementu. Obsah tohto elementu tvorí banner zaberajúci 25% obrazovky zhora, jeho obrázok pozadia a zvyšok obrazovky tvorí element typu `ScrollView`. Tento element je špecifický tým, že v prípade, že sa celý jeho obsah nemôže zobraziť na ploche, ktorá mu bola poskytnutá, umožní posúvať zobrazenie tak, aby užívateľ postupne prehliadal celý obsah. Samotné usporiadanie elementov v tomto posúvacom zobrazení je riešené formou `StackLayout`, ktorý ukladá svoj obsah za sebou v horizontálnom alebo vertikálnom smere. Sekcia s rýchlym prístupom je riešená ako mriežka, teda element typu `Grid`. Ukážka návrhu domovskej obrazovky je na obrázku 19 v kapitole 5.

Navigácia v aplikácii

Navigácia v rámci aplikácie je riešená formou hotového riešenia *MasterDetail* z knižnice *Xamarin.Forms*. Jedná sa o bočné vysúvacie menu, ktoré je možné vyvolať z ktorejkoľvek obrazovky a zobrazuje zoznam dostupných obrazoviek. Trieda obrazovky `SideMenuNavigation` rozširuje triedu `MasterDetailPage` a má za úlohu definovať, ktorá obrazovka bude domovská, čo je v tomto prípade obrazovka `HomePage`. Tiež definuje správanie po kliknutí do menu a odkazuje na obrazovku `SideMenuNavigationMaster`, ktorá definuje výzor vysúvacieho menu a jednotlivé položky menu. Generovanie potrebného kódu pre vysúvacie menu prebehlo vo vývojovom nástroji *Visual Studio* automaticky, boli potrebné len malé zmeny v dizajne a tiež zozname položiek menu. Tieto položky sú reprezentované inštanciami triedy `SideMenuNavigationMenuItem` s vlastnosťami `Title` - názov položky, `IconPath` - názov obrázku ikony a `TargetType` - typ obrazovky, na ktorú sa položka odkazuje.

Novinky

Obrazovka s novinkami je rozdelená do troch záložiek podľa jednotlivých kategórii - novinky, nadchádzajúce udalosti a ponuka práce. Pre dosiahnutie efektu záložiek musí trieda obrazovky `NewsPage` rozširovať triedu `TabbedPage`. To nám umožní v *.xaml* súbore definovať viacero ele-

[11] `https://grialkit.com/gorilla-player/`

mentov `ContentPage` a v nich dizajn stránok v jednotlivých záložkách. Každá záložka môže mať definovaný vlastný nadpis a ikonu.

Novinky sa zobrazujú v chronologickom poradí od najnovšej a pre ich zobrazenie sa využíva prvok s názvom `ListView`, teda zoznam. Dáta obsiahnuté v zozname musia byť v kolekcii, ktorá implementuje rozhranie `IEnumerable`. Kolekcia je potom pomocou atribútu `ItemsSource` nastavená ako zdroj údajov daného zoznamu. Aby sa zmeny položiek v kolekcii prejavovali v reálnom čase aj v zozname, musel som použiť kolekciu typu `ObservableCollection`.

```
private async void ListView_news_ItemAppearing(object sender,
    ItemVisibilityEventArgs e)
{
    if(((Report)e.Item).ReportId == NewsCollection.Last().ReportId) {
        CurrentOffsetNews += 10;
        NewsParams.Remove(NewsParams.Where(x => x.Key == "offset").
            FirstOrDefault());
        NewsParams.Add(new KeyValuePair<string, string>("offset",
            CurrentOffsetNews.ToString()));
        await LoadNewsAsync();
    }
}
```

Výpis 7: Implementácia metódy reagujúcej na zobrazovanie položiek zoznamu noviniek

Pretože načítavanie všetkých noviniek naraz by bolo časovo náročné a v prípade zobrazenia len jednej záložky aj zbytočné, začnú sa novinky príslušnej kategórie sťahovať až keď sa užívateľ prepne do danej záložky. To je riešené formou vyvolania udalosti `Appearing(object, EventArgs):void` na danej inštancii `ListView` pri jej zobrazení. Pri zobrazení záložky sa načíta prvých 10 noviniek. Keď sa užívateľ presunie postupne až na poslednú načítanú novinku, spustí sa načítanie ďalších 10. To je možné vďaka URL parametru `offset`, ktorý sa zvyšuje o 10 pri každom ďalšom volaní služby o vrátanie noviniek. To, či sa momentálne zobrazuje posledná načítaná novinka zisťuje metóda `ItemAppearing(object, ItemVisibilityEventArgs):void` vyvolávaná vždy, keď sa užívateľ presunie na zatiaľ nezobrazenú položku zoznamu. V metóde sa overuje, či položka zoznamu, ktorá metódu vyvolala, je zároveň poslednou v zozname všetkých položiek. Ak áno, načítajú sa ďalšie položky. Implementácia metódy je na výpise 7.

6.5.1 Jazyk užívateľského rozhrania

Keďže cieľovú skupinu používateľov tvoria aj zahraniční študenti, je potrebné, aby užívateľské rozhranie bolo lokalizované do viacerých jazykov. V prípade tejto aplikácie som implementoval rozhranie v slovenskom, českom a anglickom jazyku. To znamená, že pre každý textový reťazec, zobrazený v aplikácii musia existovať až 3 preklady. Na to slúžia *.resx* súbory, kde sú tieto

reťazce definované a v zdrojovom kóde je namiesto konkrétneho slova použitá referencia do tohto súboru. Pre každý jazyk existuje jeden *.resx* súbor. Pre správu prekladov jednotlivých reťazcov som využíval doplnok do vývojového prostredia s názvom *ResX Manager*.

6.6 Použité knižnice

Všetky knižnice použité počas vývoja tejto aplikácie boli začleňované do kódu pomocou tzv. *NuGet* balíčkov. Jedná sa o moderný spôsob zdieľania a používania knižníc hotových užitočných riešení. Všetok potrebný kód je zbalený do jedného balíčku, ktorý obsahuje skompilovaný kód v *.dll* formáte a tiež ostatné podporné súbory. Balíček je následne vystavený na verejnom alebo súkromnom úložisku, odkiaľ ho je možné stiahnuť a začleniť do svojho projektu. [14] Všetky balíčky, použité v tejto implementácii pochádzajú z verejného oficiálneho serveru *nuget.org*.

```
private void Current_ConnectivityChanged(object sender,
    ConnectivityChangedEventArgs e)
{
    if (CrossConnectivity.Current.IsConnected) {
        Frame_noConnection.IsVisible = false;
        Button_searchContact.IsEnabled = true;
    }
    else {
        Frame_noConnection.IsVisible = true;
        Button_searchContact.IsEnabled = false;
    }
}
```

Výpis 8: Udalosť vyvolaná pri strate alebo obnovení internetového pripojenia

6.6.1 Plugin Connectivity

Táto knižnica sa využíva pre podporu práce s pripojením. Využíval som ju pre zistenie, či je zariadenie pripojené k internetu, aby sa zbytočne nespúšťala sieťová komunikácia, ak by pripojenie nebolo dostupné. Knižnica tiež poskytuje zistenie aktuálneho typu pripojenia. To som využil pre implementáciu ochrany, aby sa synchronizácia nevykonala pri dátovom pripojení, ak si to užívateľ tak nastaví. Pre detekciu výpadku alebo obnovenia pripojenia som použil príslušné udalosti, z tejto knižnice. Ak napríklad pri zobrazení obrazovky z novinkami nie je zariadenie online, novinky sa začnú načítavať okamžite po nadviazaní spojenia.

Na výpise 8 je zobrazené použitie udalosti pri strate alebo obnovení pripojenia na obrazovke vyhľadávania kontaktov. Pri strate pripojenia sa zobrazí na obrazovke upozornenie a zakáže sa tlačidlo pre spustenie vyhľadávania. Pri obnovení pripojenia sa upozornenie skryje a tlačidlo povolí.

6.6.2 Plugin Settings

Knižnica, umožňujúca prístup k užívateľským dátam aplikácii. Jedná sa o dáta typu *kľúč-hodnota*, ktoré zostanú uložené aj po ukončení aplikácie, no zároveň sú dostupné rýchlejšie, než keby boli uložené v databáze. Toto úložisko je pre danú aplikáciu načítané celé pri každom spustení aplikácie, preto by nemalo byť zbytočne zahlcované, aby sme zachovali rýchle načítanie aplikácie.

Pomocou knižnice som ukladal do užívateľských dát napríklad prístupový token pre volanie API, dátum a čas poslednej synchronizácie databázy alebo príznak, či sa aplikácia spúšťa prvý krát. Implementácia ukladania a získavania prístupového tokenu je na výpise 9.

```
private static ISettings AppSettings {
    get {
        return CrossSettings.Current;
    }
}
private const string ApiTokenKey = "api_token";
private static readonly string ApiTokenDefault = string.Empty;
public static string ApiToken {
    get {
        return AppSettings.GetValueOrDefault(ApiTokenKey, ApiTokenDefault);
    }
    set {
        AppSettings.AddOrUpdateValue(ApiTokenKey, value);
    }
}
```

Výpis 9: Ukážka práce s knižnicou *PluginSettings*

6.6.3 Newtonsoft JSON

Jedná sa o jednu z najrýchlejších knižníc určených, pre prácu s dátovým formátom *JSON*. Obsahuje sadu tried, podporujúcich efektívnu a rýchlu konverziu medzi .NET objektami a ich ekvivalentom v JSON formáte. [12] Pri implementácii som najčastejšie využíval triedu `JsonConvert`, ktorá obsahuje potrebné metódy pre konverzie. Na výpise 1 je implementovaná metóda pre získanie tokenu. Dáta z API sú získané vo formáte JSON a pre ďalšie použitie je potrebné ich previesť do inštancie triedy `TokenInfo` pomocou statickej metódy `DeserializeObject<T>(string):T`.

6.6.4 System.Net.Http

Poskytuje rozhranie, ktoré dovoľuje aplikáciam spracovávať webové služby cez HTTP protokol a HTTP komponenety, využívané ako klientami, tak aj servermi. [15] Obsahuje napríklad

triedy pre prácu s HTTP požiadavkami a odpoveďami, výnimky alebo typy prenášaných dát. V tejto aplikácii bola využitá trieda `HttpClient`, ktorá obsahuje metódy dovoľujúce odosielať požiadavky a prijímať odpovede cez HTTP protokol. [16] Trieda sa využíva napríklad pri implementácii autorizácie, synchronizácie alebo sťahovaniu univerzitných noviniek. Na výpise 1 je ukážka použitia asynchrónnej metódy typu POST triedy `HttpClient`.

```
public T Select<T>(object primaryKey) where T : class, new()
{
    using (SQLiteConnection conn = DependencyService.Get<ISQLiteDb>().
        GetConnection()) {
        try {
            return conn.GetWithChildren<T>(primaryKey);
        }
        catch (Exception) {
            return default(T);
        }
    }
}
```

Výpis 10: Použitie triedy `SQLiteconnection` pre získanie dát z tabuľky

6.6.5 SQLite.Net-PCL

Knižnica podporujúca prácu s databázou typu SQLite 3. Jedna z najdôležitejších knižníc v tejto aplikácii. Umožňuje implementáciu rýchlej databázovej vrstvy systému a poskytuje jednoduché metódy pre operácie s databázou. [17] Pre pripojenie k databáze som využíval triedu `SQLiteConnection`, ktorá implementuje metódy napríklad pre prácu s transakciami, vytváranie a úpravu tabuliek, operácie s dátami tabuliek a iné. Každá služba API má vlastnú triedu, ako nosič dát. Pre vytvorenie tabuľky s primárnym kľúčom som danú vlastnosť triedy označil atribútom `PrimaryKey`. Namiesto názvov tabuliek sa využívajú generické parametre, na základe ktorých sa použije alebo vytvorí požadovaná tabuľka. Na výpise 10 je ukážka práce s databázovým pripojením pre získanie dát z tabuľky. Je použitý konštrukt `using` pre bezpečné ukončenie práce s pripojením a jeho následné uvoľnenie. Pre získanie dát z tabuľky je použitá metóda `GetWithChildren<T>(object):T`, ktorá zároveň získa aj vnorené objekty daného typu.

6.6.6 Xamarin.Forms.Maps

Táto knižnica využíva natívne API máp danej platformy. Pre každú platformu je nutné vykonať pár inicializačných krokov, no výsledkom bude plná integrácia máp do spoločného kódu pre všetky platformy. To znamená, že s natívnym API máp bude možné pracovať ako s prvkom z knižnice *Xamarin.Forms*. [18]

Na výpise 11 je zobrazený konštruktor pre obrazovku s mapou, ktorý ako parametre prijíma súradnice bodu a text, ktorým bude tento bod označený. V konštruktore sa vytvorí nová inštancia triedy `Map` z danej knižnice, nastaví sa jej stred zobrazenia na zadané súradnice a zvolí sa hybridný typ mapy. Potom sa vytvorí inštancia triedy `Pin`, predstavujúca bod, ktorej sa nastavia súradnice a štítok. Bod sa potom vloží do mapy a celá mapa sa následne vloží do koreňového elementu celej obrazovky, ktorý ho zobrazí.

```
public MapPage(double latitude, double longitude, string label)
{
    InitializeComponent();
    Map Map_campusMap = new Map();
    Map_campusMap.MoveToRegion(MapSpan.FromCenterAndRadius(new Position(
        latitude, longitude), Distance.FromMiles(0.5)));
    Map_campusMap.MapType = MapType.Hybrid;
    Pin pin = new Pin() {
        Position = new Position(latitude, longitude),
        Label = label
    };
    Map_campusMap.Pins.Add(pin);
    StackLayout_mapPage.Children.Add(Map_campusMap);
}
```

Výpis 11: Ukážka práce s knižnicou pre mapy

7 Testovanie

Samozrejmosťou pri vývoji aplikácie je jej samotné testovanie. Na to som využíval ako fyzické, tak aj virtuálne zariadenia. Keďže aplikácia komunikuje s RESTful API, do procesu vývoja bolo potrebné zahrnúť aj testovanie samotného API.

7.1 Problémy s RESTful API

V prípade API pre kontaktné informácie som zaznamenal chybu, kedy získané údaje vo formáte vCard neobsahovali diakritické znaky, resp. boli nahradené znakom "?". Túto chybu som oznámil zodpovednej osobe v CIT, ktorá ju následne opravila. Ďalší nedostatok, ktorý som zaznamenal bol v prípade anglického prekladu názvov alebo popisov miest v API navigácie. Text nie je vôbec preložený, je v českom jazyku a s predponou "ENG:". Tento nedostatok som taktiež oznámil.

Na testovanie služieb API som využíval softvér s názvom *Postman*[12]. Používal som základnú bezplatnú licenciu pre individuálnych vývojárov s možnosťou zasielania 1000 žiadostí do API mesačne. Jedná sa o kvalitný nástroj, poskytujúci širokú paletu nastavení volaných metód.

7.2 Testovanie aplikácie

Aplikáciu som testoval na fyzických a virtuálnych zariadeniach rôznych veľkostí a rozlíšení obrazoviek, aby som overil, že aplikácia sa bude správne zobrazovať na všetkých typoch obrazoviek.

Android

Súčasťou inštalácie vývojových nástrojov Android SDK je aj nástroj *Android Virtual Device Manager*. Ten umožňuje vytváranie virtuálnych Android zariadení rôznych typov, napríklad mobilný telefón, tablet, Android hodinky alebo Android televízia. K tomu je potrebné mať stiahnutý obraz systému požadovanej verzie. Virtualizované zariadenia však nedosahujú takú odozvu systému a taký výkon, ako fyzické.

Pri využívaní fyzických zariadení bolo potrebné povoliť režim ladenia, čiže povoliť rozhranie *Android Debug Bridge*. Následné bolo možné sa pripojiť k vývojovému prostrediu buď pomocou kábla alebo cez WiFi pomocou protokolu TCP/IP.

Testovanie prebiehalo na fyzických zariadeniach s verziou systému Android 5.1.1, 6.0.1, 7.1, 8.1, a na virtuálnych s verziou 4.4 a 6.0.

iOS

Na kompiláciu aplikácii pre platformu iOS je potrebná sada nástrojov iOS SDK a vývojové prostredie *Apple Xcode*. Tie je však možné nainštalovať len na zariadenia s operačným systémom Mac OS, teda len na zariadenia od firmy Apple. Súčasťou nástrojov sú tiež virtuálne zariadenia

[12] `https://www.getpostman.com/`

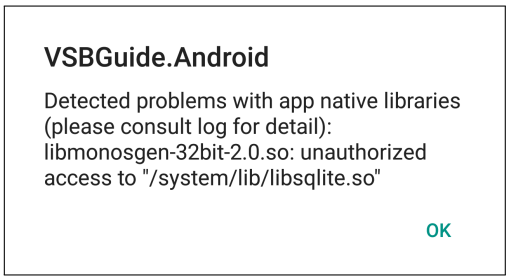
iPhone a iPad. Tie sú automaticky vytvorené a spustené po kompilácii zdrojového kódu a následne je na nich aplikácia spustená. Virtuálne zariadenia sa vždy spustia s najnovšou vydanou verziou systému, takže som aplikáciu testoval na verzii 11.2 a po vydaní novej aktualizácie na 11.3.

Keďže počas vývoja tejto aplikácie som nevlastnil zariadenie Apple, bol mi so strany univerzity poskytnutý prístup ku školským iMac zariadeniam. Na nich som priebežne testoval aj iOS verziu aplikácie.

7.2.1 Problémy s databázou

V prípade platformy Android som pri testovaní aplikácie zaznamenal problémy pri práci s databázou. Tento problém sa prejavoval varovným dialógovým oknom, ktorý sa zobrazoval pri opakovanom spustení aplikácie. Dialógové okno je zobrazené na obrázku 26. Práca s databázou však nebola obmedzená a dáta bolo možné bez problémov vkladať alebo získavať.

Po dlhom skúmaní rôznych fór som sa dozvedel, že výrobca daného operačného systému, Google, od verzie Android 7.0, zamedzil určitým existujúcim knižniciam priamy prístup do databázy, nakoľko hrozilo poškodenie uložených dát. Riešením, ku ktorému som sa dostal, bolo nahradenie triedy `SQLitePlatformAndroid` za triedu `SQLitePlatformAndroidN`. Tá sa nachádzala v ďalšom *nuget* balíčku, ktorý som musel stiahnuť dodatočne. Po danom kroku sa už upozornenie cez dialógové okno nezobrazuje.



Obr. 26: Varovné dialógové okno s popisom chyby s natívnymi knižnicami

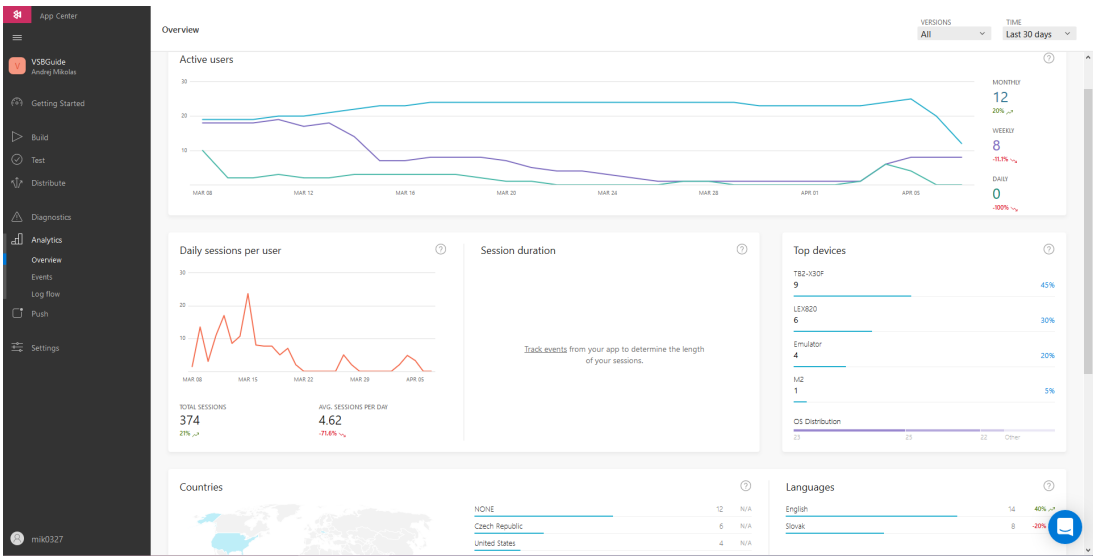
7.3 Zisťovanie ďalších chýb

Pre prípad, že by užívateľ počas používania aplikácie narazil na nejaký nedostatok, vybavil som aplikáciu aj možnosťou odoslania správy o chybe. Nachádza sa na obrazovke s nastaveniami v sekcii so spätnou väzbou. Po kliknutí na túto možnosť je užívateľ presmerovaný do jeho vstavaného mailového klienta, kde sa zobrazí okno s novou správou s prednastaveným mailom príjemcu a predmetom správy. Tam môže užívateľ popísať, aký problém s aplikáciou zaznamenal, prípadne priložiť snímky obrazovky.

7.3.1 Visual Studio App Center

Pre ďalšie vylepšovanie a skvalitňovanie aplikácie som zakomponoval možnosť zbierania anonymných štatistických údajov a chybových hlásení. Tieto údaje v žiadnom prípade nie je možné použiť na určenie identity používateľa, o čom je užívateľ informovaný pri prvom spustení aplikácie. Slúži na to nástroj *Visual Studio App Center* a knižnice *Microsoft.AppCenter*, *Microsoft.AppCenter.Analytics* pre analytické údaje a *Microsoft.AppCenter.Crashes* pre chybové hlásenia. Služba sa spúšťa v metóde `OnStart():void` v spúšťacej triede celej aplikácie `App`. Aplikácia je identifikovaná jednoznačným kľúčom, ktorý vygenerovalo webové rozhranie tohto nástroja. Pre každú platformu je potrebné v nástroji vytvoriť nový projekt a vygenerovať identifikátor. Všetky zozbierané údaje sa zobrazujú vo webovom rozhraní, ktorého ukážka je na obrázku 27. Štatistické údaje zahŕňajú:

- Počet aktívnych užívateľov aplikácie za určité obdobia
- Priemerný denný počet spustení aplikácie jedným užívateľom
- Trvanie relácií
- Štatistiky zariadení, využívajúce aplikáciu
- Štatistky štátov, v ktorých užívatelia aplikáciu používajú
- Štatistiky jazykov zariadení, na ktorých je aplikácia spúšťaná



Obr. 27: Prehľad používania aplikácie vo webovom rozhraní *Visual Studio App Center*

Chybové hlásenia obsahujú kompletný chybový log, vygenerovaný aplikáciou pri páde. Na základe nich je potom možné chybu analyzovať a odstrániť, aby k ďalším pádom nedochádzalo.

Ku každej chybe je tiež uvedený počet užívateľov, ktorí túto chybu zaznamenali, verzia aplikácie, na ktorej nastala chyba a približný čas pádu aplikácie.

Ďalšou možnosťou ladenia aplikácie pomocou tohto nástroja je odosielanie informácii o výnimkách, ktoré nastali počas behu aplikácie, ale nespôsobili jej pád. Jedná sa teda o výnimky ošetrené v `try-catch` blokoch. Slúži na to statická metóda `TrackError(Exception):void` triedy `Crashes`, ktorej sa do parametru predá vyvolaná výnimka. Ukážka použitia metódy je na výpise 1 v kapitole 6.

Záver

Praktickým výstupom tejto práce mal byť návrh a implementácia študentskej aplikácie pre VŠB-TUO. Tento cieľ sa mi podarilo splniť. Aplikácia v súčasnom stave poskytuje relevantné informácie o kontaktných údajoch zamestnancov, univerzitných novinkách a tiež umožňuje základnú orientáciu v areáloch univerzity. Aplikácii som dal príznačný názov "*VŠB Guide*", ktorý vznikol zložením skratky názvu univerzity "VŠB" a slova "guide", ktoré v anglickom jazyku znamená "sprievodca" alebo "príručka". Jedná sa teda o "sprievodcu po VŠB".

RESTful API poskytované CIT spočiatku neposkytovalo získavanie informácii o rozvrhu a informácii o štúdiu. Tieto funkcie boli navrhnuté až na moju žiadosť, no ich implementácia bola dokončená približne 3 týždne pred termínom odovzdania tejto práce. Z tohto dôvodu sú tieto funkcionality zahrnuté v kapitole návrhu aplikácie avšak v implementačnej časti už nie. Do samotnej aplikácie ich však plánujem zahrnúť v čo najkratšom čase, nakoľko som sa primárne zameriaval na doladenie súčasného stavu aplikácie. Následne mám v pláne zverejniť ju na distribučných platformách *Google Play Store* a *Apple App Store*, odkiaľ bude zdarma dostupná na stiahnutie pre študentov a návštevníkov VŠB-TUO.

Z vykonaného prieskumu trhu, som zistil, že v súčastnosti na poli mobilných operačných systémov zaberajú vedúce postavenia OS Android a iOS. Taktiež som zistil, že v prípade výkonovo nenáročných aplikácii pre viac platforiem je výhodnejšie vydať sa cestou multiplatformového vývoja, konkrétne hybridného spôsobu. Preto tento prístup bol aplikovaný aj v prípade tejto práce.

Pri porovnaní aplikácii iných univerzít s mojim riešením by som vyzdvihol možnosť zobrazovať univerzitné novinky, ktorú iné aplikácie neposkytovali. Na druhej strane, všetky porovnávané aplikácie obsahovali rozvrh študenta, ktorý však moja aplikácia zatiaľ nemá. V porovnaní s existujúcimi aplikáciami určených pre študentov tejto univerzity môžem na základe vlastného názoru povedať, že moje riešenie vyniká nad ostatnými ako po grafickej, tak aj funkčnej stránke. V neposlednom rade mojej aplikácii pridáva na kvalite aj fakt, že je určená pre dve súčasne najrozšírenejšie mobilné platformy a na oboch z nich je plne funkčná, na rozdiel od pár popisovaných aplikácii.

Súčasné služby CIT pre účely orientácie sú podľa môjho názoru na dobrej úrovni. Poskytujú jednoduchú orientáciu buď pomocou vlastných máp areálov, ktoré sú dostupné online a ako navigačné tabule v areáloch, alebo pomocou upravených máp od spoločnosti Google. Služby RESTful API od CIT poskytujú relevantné informácie použiteľné pre potreby orientácie v areáloch, získavania kontaktných informácii a univerzitných noviniek. Na moju žiadosť je po doplnení funkcionality tiež možné získavať rozvrhové informácie a informácie o štúdiu. RESTful API považujem za veľmi efektívny a pohodlný spôsob poskytovania funkcionality webového informačného systému, so širokými možnosťami využitia.

Proces návrhu aplikácie je veľmi dôležitý krok v procese vývoja softvéru, nakoľko si pri ňom uvedomíme, čo všetko od systému požadujeme a akým spôsobom tieto požiadavky budeme riešiť.

Významným krokom bolo určenie minimálnych podporovaných verzií cieľových platforiem, ktoré mohlo značne ovplyvniť aj výber frameworku použitého pre implementáciu. Diagram komponent mi dal jasnejšiu predstavu, do akých modulárnych jednotiek musím aplikáciu rozčleniť a ako jednotlivé vrstvy aplikácie budú medzi sebou komunikovať pomocou rozhraní.

Počas implementácie som musel vyriešiť množstvo implementačných problémov, najmä čo sa týka práce s databázou a jej synchronizácia s RESTful API, práce s pripojením na internet alebo zobrazovaním máp. Implementáciu mi však veľmi uľahčili hotové riešenia v podobe knižníc, ktoré som do projektu zakomponoval. Pri implementácii užívateľského rozhrania som sa snažil aby sa prvky na obrazovke zobrazovali korektne a prehľadne na rôznych veľkostiach cieľových zariadení. Počas testovania som overoval funkčnosť aplikácie na oboch platformách a na rôznych verziách týchto platforiem.

Vývoju a tejto aplikácie mám záujem sa venovať aj naďalej, rozširovať jej funkcionalitu a zdokonaľovať súčasný stav. Verím, že v spolupráci s CIT bude neskôr možné do aplikácie zahrnúť napríklad možnosť prihlasovania sa na termíny skúšok, prezerať študijné výsledky, zobraziť harmonogram semestra alebo integrovať mailového klienta priamo do aplikácie. V prípade, že by aplikácia zaznamenala medzi študentami úspech, pridal by som ďalšie jazyky pre študentov študijného programu *Erasmus*. V súčasnosti prechádzajú webové stránky univerzity na nový vizuálny štýl, preto by po jeho doladení a ustálení mohla byť aj táto aplikácia prispôsobená tomuto štýlu. Taktiež nevylučujem možnosť vytvoriť aj verziu aplikácie pre pedagógov.

Literatúra

[1] *UPlikace* [online]. [cit. 2017-12-15]. Dostupné z: https://play.google.com/store/apps/details?id=cz.uplikace.app.

[2] *Muni IS* [online]. [cit. 2017-12-15]. Dostupné z: https://play.google.com/store/apps/details?id=cz.krupka.muniis.

[3] *Student ZČU* [online]. [cit. 2017-12-15]. Dostupné z: https://play.google.com/store/apps/details?id=com.tommylabs.portalzcu.

[4] *Virtual FIIT* [online]. [cit. 2017-12-18]. Dostupné z: https://play.google.com/store/apps/details?id=sk.stuba.fiit.virtfiit.

[5] *AIS2 študent* [online]. [cit. 2017-12-18]. Dostupné z: https://play.google.com/store/apps/details?id=ais.mais.android.client.terminy.

[6] *Průvodce VŠB-TU Ostrava* [online]. [cit. 2018-04-06]. Dostupné z: https://play.google.com/store/apps/details?id=com.effortix.app110.

[7] *VŠB Telefonní Seznam* [online]. [cit. 2018-04-06]. Dostupné z: https://play.google.com/store/apps/details?id=cz.vsb.tuo.vsbtelefonniseznam.

[8] *Rozvrh VŠB-TUO* [online]. [cit. 2018-04-06]. Dostupné z: https://play.google.com/store/apps/details?id=name.kocian.vsb.timetable.

[9] *VŠB Navi* [online]. [cit. 2018-04-06]. Dostupné z: https://play.google.com/store/apps/details?id=cz.vsb.vsbnavi.

[10] LIČKO, Ondrej. *Téma týždňa: Mobilné operačné systémy a ich porovnanie* [online]. 2013 [cit. 2018-04-06]. Dostupné z: https://www.mojandroid.sk/tema-tyzdna-mobilne-operacne-systemy-a-ich-porovnanie/.

[11] *Mapa areálu* [online]. [cit. 2017-12-20]. Dostupné z: https://www.vsb.cz/cs/o-univerzite/kontakty-mapy-parkovani/mapy-arealu.

[12] *Serializing and Deserializing JSON* [online]. [cit. 2018-03-26]. Dostupné z: https://www.newtonsoft.com/json/help/html/SerializingJSON.htm.

[13] BRITCH, David. DUNN, Craig. *Xamarin.Forms Requirements* [online]. 2017 [cit. 2018-03-26]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/get-started/installation?tabs=vswin.

[14] BROCKSCHMIDT, Kraig. MYERS, Alfred. *An introduction to NuGet* [online]. 2018 [cit. 2018-03-26]. Dostupné z: https://docs.microsoft.com/en-us/nuget/what-is-nuget.

[15] *System.Net.Http* [online]. 2017 [cit. 2018-03-26]. Dostupné z: https://www.nuget.org/packages/System.Net.Http/.

[16] *HttpClient Class* [online]. [cit. 2018-03-26]. Dostupné z: https://msdn.microsoft.com/library/system.net.http.httpclient(v=vs.110).aspx.

[17] *oysteinkrog/SQLite.Net-PCL* [online]. 2017 [cit. 2018-03-27]. Dostupné z: https://github.com/oysteinkrog/SQLite.Net-PCL.

[18] BRITCH, David. DUNN, Craig. UMBAUGH, Brad. *Map* [online]. 2016 [cit. 2018-03-27]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/map.

[19] *Class diagram - diagram tříd* [online]. [cit. 2018-03-27]. Dostupné z: http://mpavus.wz.cz/uml/uml-s-class-3-3-1.php.

[20] ČÁPKA, David. *5. díl - UML - Class diagram* [online]. [cit. 2018-03-27]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model.

[21] ČÁPKA, David. *4. díl - UML - Doménový model* [online]. [cit. 2018-03-28]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-domenovy-model-diagram.

[22] REJNKOVÁ, Petra. *Diagram tříd* [online]. [cit. 2018-03-28]. Dostupné z: http://uml.czweb.org/diagram_trid.htm.

[23] REJNKOVÁ, Petra. *Diagram nasazení* [online]. [cit. 2018-03-28]. Dostupné z: http://uml.czweb.org/diagram_nasazeni.htm.

[24] REJNKOVÁ, Petra. *Diagram komponent* [online]. [cit. 2018-03-28]. Dostupné z: http://uml.czweb.org/diagram_komponent.htm.

[25] *Xamarin vs Apache Cordova: Which One Is Better?* [online]. 2017 [cit. 2018-03-30]. Dostupné z: https://skelia.com/articles/xamarin-vs-apache-cordova-one-better/.

[26] *Apache Cordova vs. Xamarin* [online]. [cit. 2018-03-30]. Dostupné z: https://stackshare.io/stackups/apache-cordova-vs-xamarin.

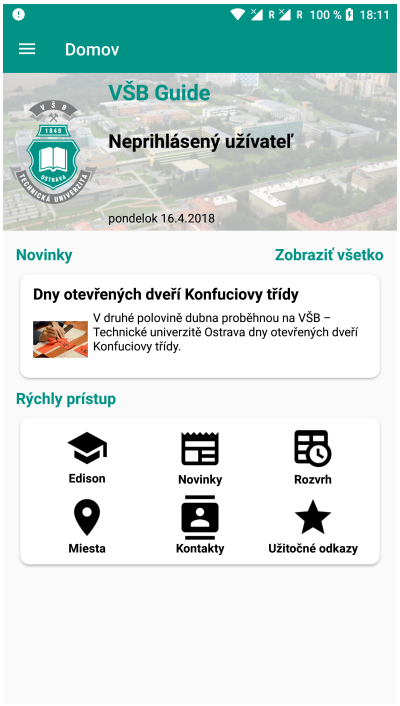
[27] *Xamarin* [online]. [cit. 2018-03-30]. Dostupné z: https://www.xamarin.com/platform.

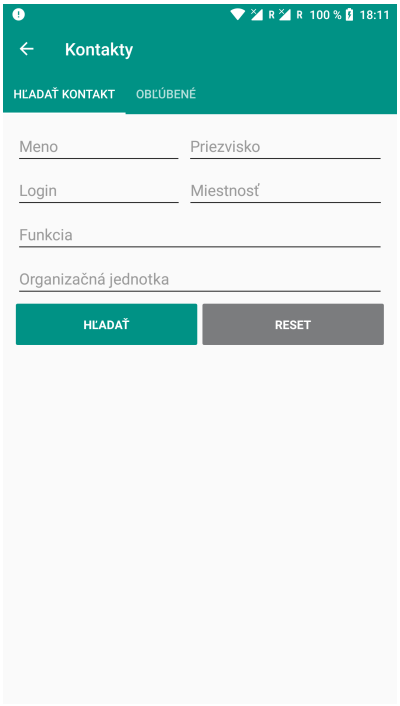
[28] MALÝ, Martin. *REST: architektura pro webové API* [online]. 2009 [cit. 2018-01-15]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/.

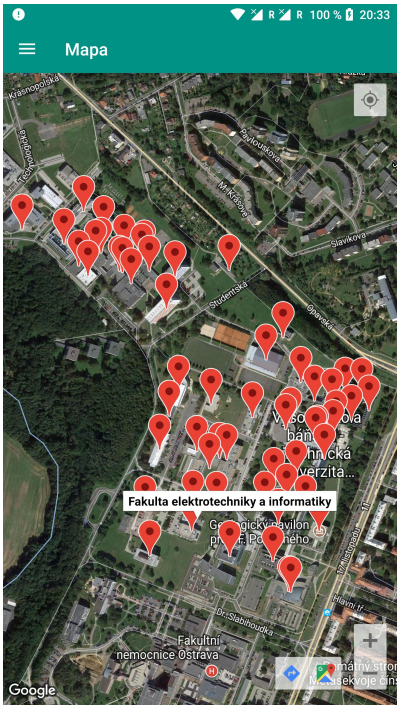
[29] Massé, M. *REST API Design Rulebook*. O'Reilly Media, Sebastopol, 2012.

A Náhľad hotového užívateľského rozhrania

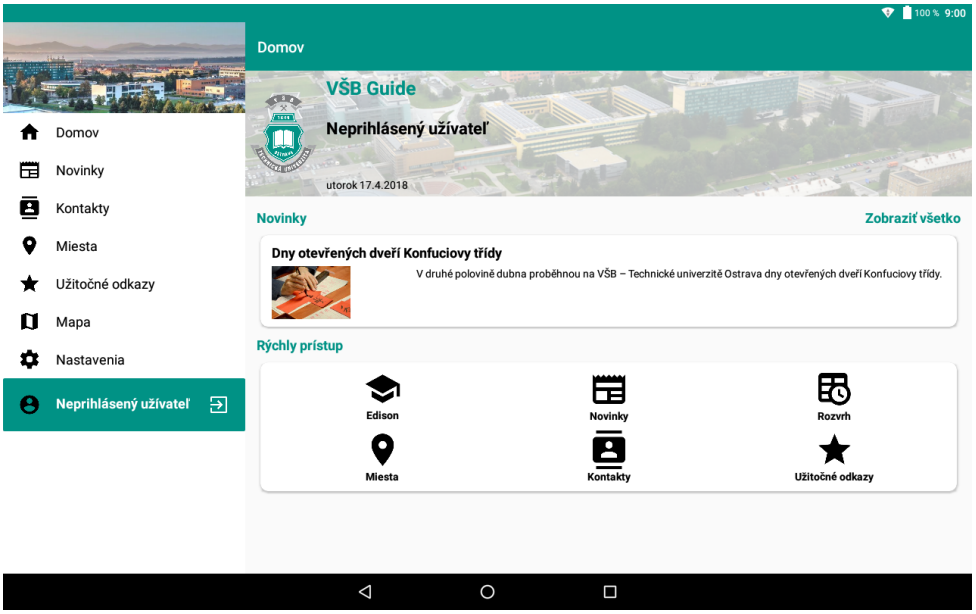
A.1 Android telefón

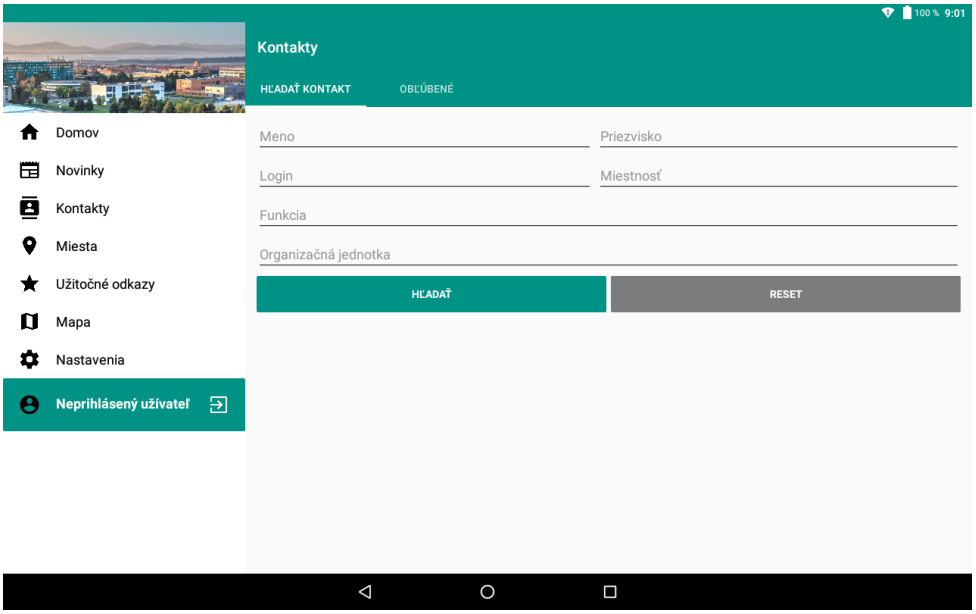


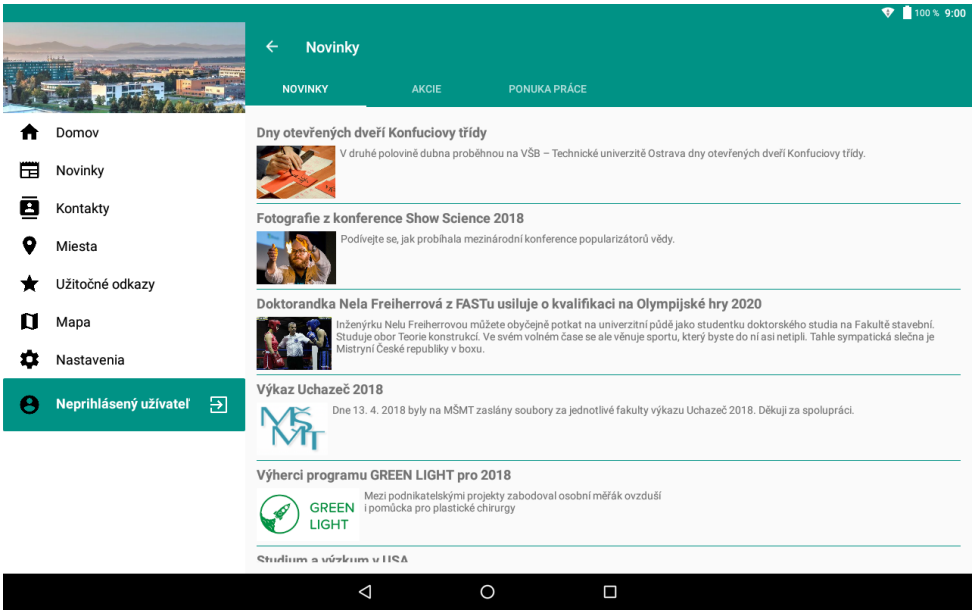


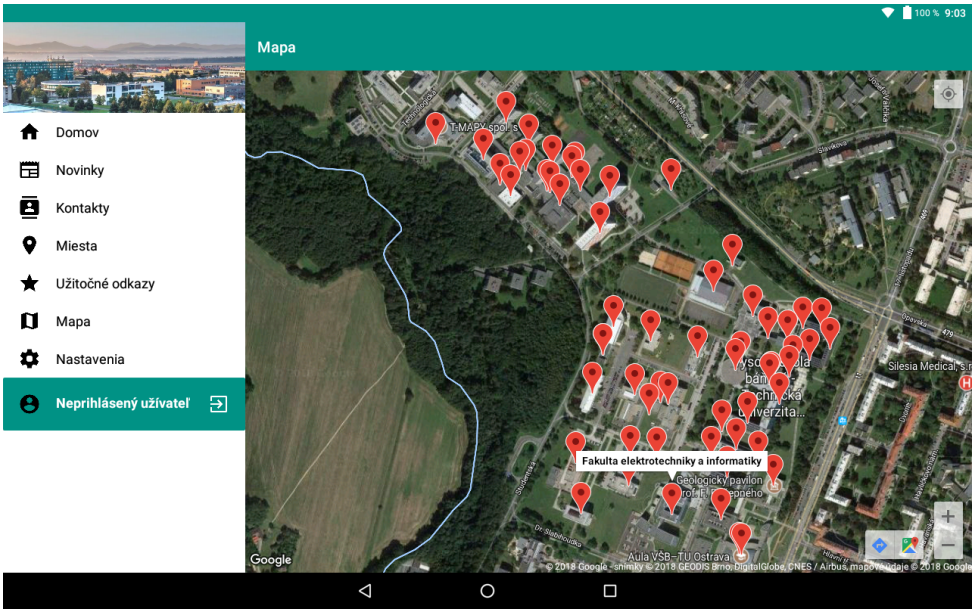


A.2 Android tablet

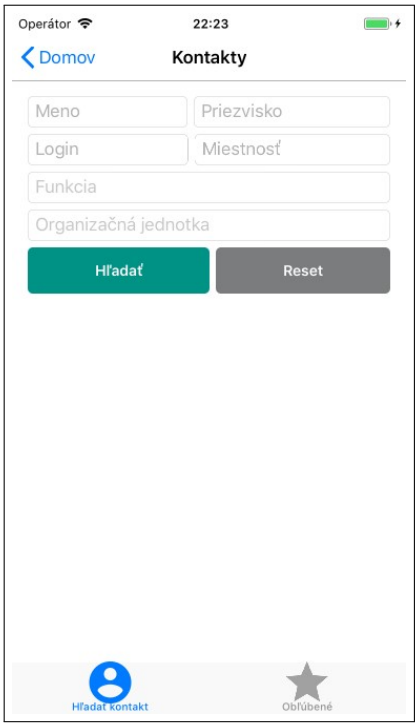


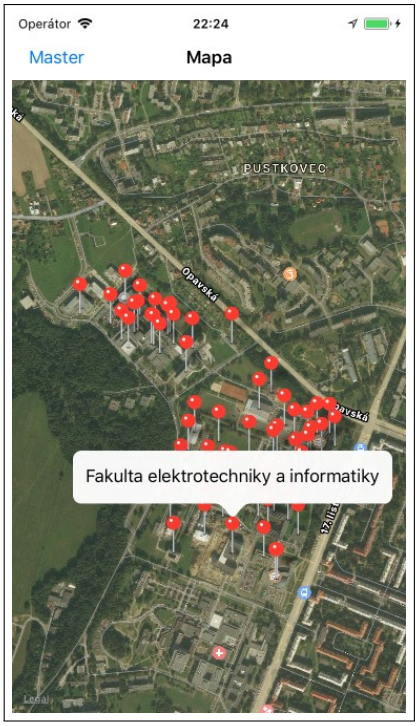



100 % 9:00
Novinky
NOVINKY
AKCIE
PONUKA PRÁCE
Domov
Novinky
Kontakty
Miesta
Užitočné odkazy
Mapa
Nastavenia
Neprihlásený užívateľ
Dny otevřených dveří Konfuciovy třídy
V druhé polovině dubna proběhnou na VŠB – Technické univerzitě Ostrava dny otevřených dveří Konfuciovy třídy.
Fotografie z konference Show Science 2018
Podívejte se, jak probíhala mezinárodní konference popularizátorů vědy.
Doktorandka Nela Freiherrová z FASTu usiluje o kvalifikaci na Olympijské hry 2020
Inženýrku Nelu Freiherrovou můžete obyčejně potkat na univerzitní půdě jako studentku doktorského studia na Fakultě stavební. Studuje obor Teorie konstrukcí. Ve svém volném čase se ale věnuje sportu, který byste do ní asi netipli. Tahle sympatická slečna je Mistryní České republiky v boxu.
Výkaz Uchazeč 2018
Dne 13. 4. 2018 byly na MŠMT zaslány soubory za jednotlivé fakulty výkazu Uchazeč 2018. Děkuji za spolupráci.
Výherci programu GREEN LIGHT pro 2018
GREEN LIGHT
Mezi podnikatelskými projekty zabodoval osobní měřák ovzduší i pomůcka pro plastické chirurgy
Studium a výzkum v USA

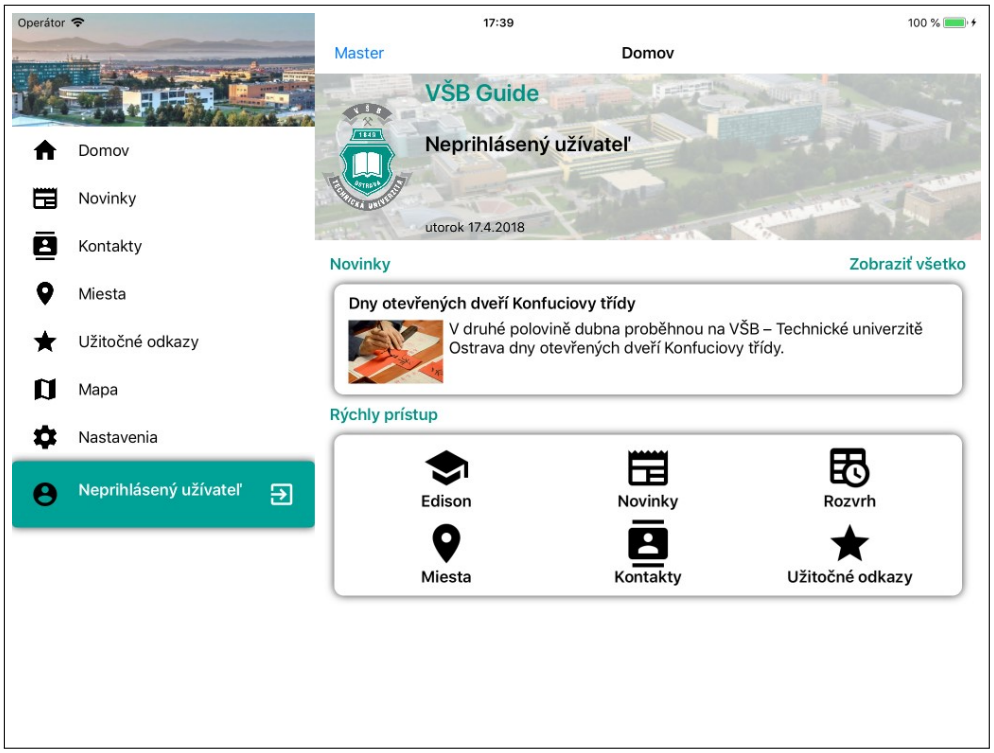

100 % 9:03
Mapa
Domov
Novinky
Kontakty
Miesta
Užitočné odkazy
Mapa
Nastavenia
Neprihlásený užívateľ
Fakulta elektrotechniky a informatiky
Google

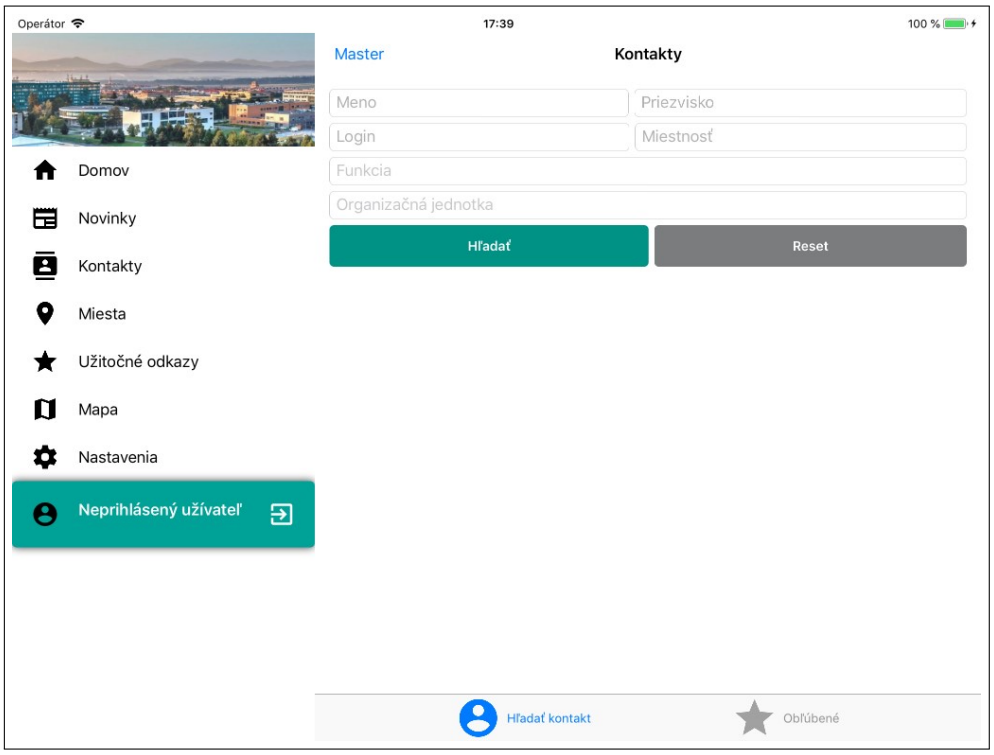
A.3 iOS telefón

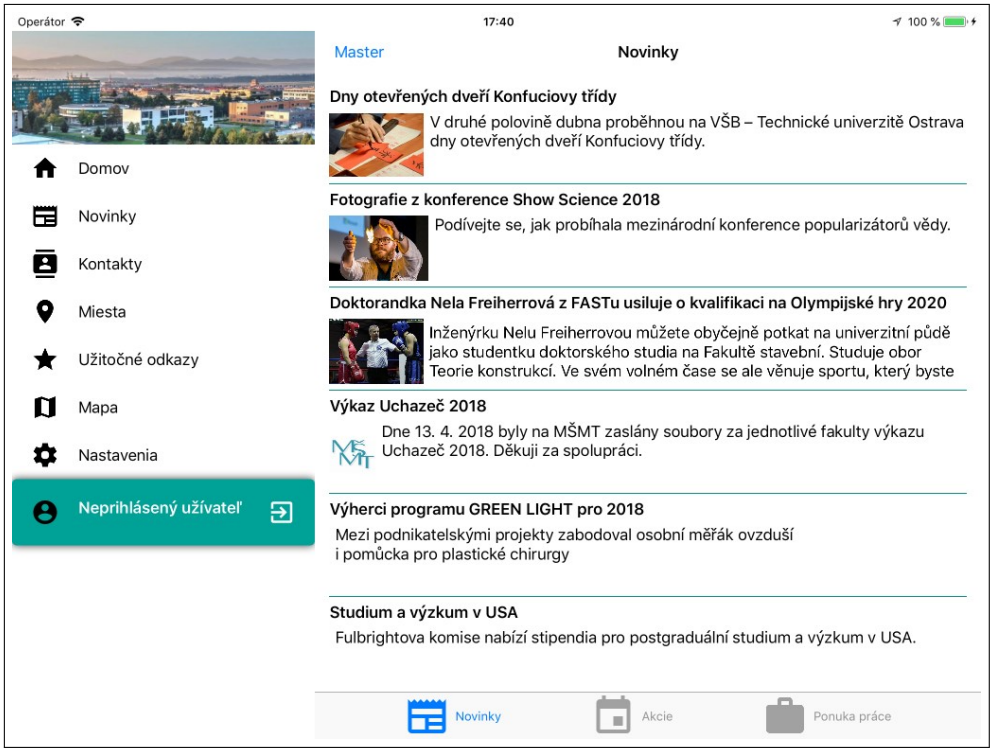


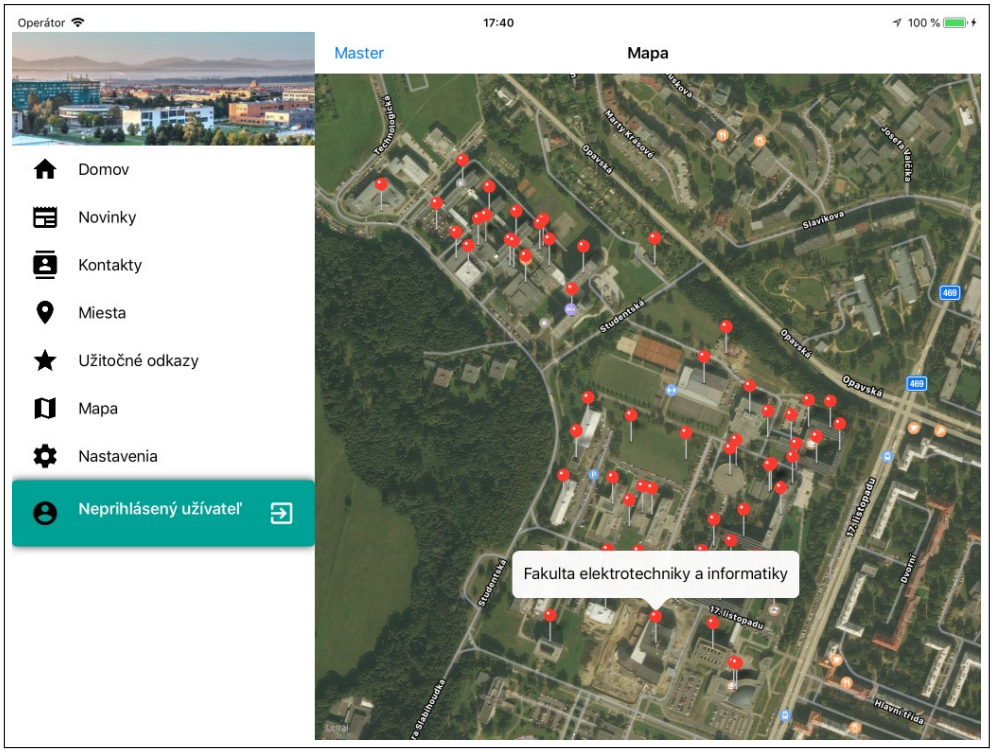


A.4 iOS tablet






Operátor
17:40
100 %
Domov
Novinky
Kontakty
Miesta
Užitočné odkazy
Mapa
Nastavenia
Neprihlásený užívateľ
Master
Novinky
Dny otevřených dveří Konfuciovy třídy
V druhé polovině dubna proběhnou na VŠB – Technické univerzitě Ostrava dny otevřených dveří Konfuciovy třídy.
Fotografie z konference Show Science 2018
Podívejte se, jak probíhala mezinárodní konference popularizátorů vědy.
Doktorandka Nela Freiherrová z FASTu usiluje o kvalifikaci na Olympijské hry 2020
Inženýrku Nelu Freiherrovou můžete obyčejně potkat na univerzitní půdě jako studentku doktorského studia na Fakultě stavební. Studuje obor Teorie konstrukcí. Ve svém volném čase se ale věnuje sportu, který byste
Výkaz Uchazeč 2018
Dne 13. 4. 2018 byly na MŠMT zaslány soubory za jednotlivé fakulty výkazu Uchazeč 2018. Děkuji za spolupráci.
Výherci programu GREEN LIGHT pro 2018
Mezi podnikatelskými projekty zabodoval osobní měřák ovzduší i pomůcka pro plastické chirurgy
Studium a výzkum v USA
Fulbrightova komise nabízí stipendia pro postgraduální studium a výzkum v USA.
Novinky
Akcie
Ponuka práce


Operátor
17:40
100 %
Domov
Novinky
Kontakty
Miesta
Užitočné odkazy
Mapa
Nastavenia
Neprihlásený užívateľ
Master
Mapa
Fakulta elektrotechniky a informatiky
Opavská
Studentská
17. listopadu
Hlavní třída
Slavíkova
Josefa Valečka
Marty Krásové
Technologická
Dvořní
Studentská
469

B Obsah CD

K práci priložené CD obsahuje nasledujúce súbory:

1. Text bakalárskej práce
    - MIK0327.pdf
2. Projekt so zdrojovými kódmi aplikácii pre Android a iOS
    - VSBGuide
3. Inštalačné súbory aplikácii pre Android a iOS
    - VSBGuide.apk
    - VSBGuide.ipa